Centro de Investigación Científica y de Educación Superior de Ensenada

SIMULACION EFICIENTE DE MODELOS PARALELOS DR-MESH EN LR-MESH

TESIS
MAESTRIA EN CIENCIAS

JOSE ANTONIO CARDENAS HARO

Ensenada, Baja California. Noviembre de 2001.

TESIS DEFENDIDA POR

José Antonio Cárdenas Haro

Y APROBADA POR EL SIGUIENTE COMITÉ

Dr. José Alberto Fernández Zepeda
Director del Comité

M. en C. José Luis Briseño Cervantes
Miembro del Comité

Dr. Andrei Tchernykh
Miembro del Comité

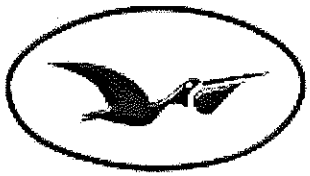
M. en C. José Douglas Frez Cárdenas
Miembro del Comité

M. en C. José Luis Briseño Cervantes
Jefe del Departamento de Ciencias de la Computación

Dr. Luis Alberto Delgado Argote
Director de Estudios de Posgrado

22 de noviembre 2001

Centro de Investigación Científica y de
Educación Superior de Ensenada



DIVISIÓN DE FÍSICA APLICADA
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

SIMULACIÓN EFICIENTE DE MODELOS PARALELOS DR-MESH EN LR-MESH

TESIS
que para cubrir parcialmente los requisitos
necesarios para obtener el grado de
MAESTRO EN CIENCIAS
presenta:

JOSÉ ANTONIO CÁRDENAS HARO

Ensenada, Baja California, México. Noviembre 2001

RESUMEN de la tesis de JOSÉ ANTONIO CÁRDENAS HARO, presentada como requisito parcial, para la obtención del grado de MAESTRO EN CIENCIAS DE LA COMPUTACIÓN. Ensenada, Baja California, México. Noviembre del 2001.

SIMULACIÓN EFICIENTE DE MODELOS PARALELOS DR-MESH EN LR-MESH

Resumen aprobado por:

Dr. José Alberto Fernández Zepeda
Director de Tesis

En este documento se examinan las características computacionales de algunos modelos paralelos con ductos reconfigurables, los cuales han mostrado ser más poderosos que los modelos convencionales como la PRAM. El poder de estos modelos está dado por su habilidad de alterar dinámicamente las conexiones entre procesadores, por ello se pueden desarrollar algoritmos rápidos para resolver una gran cantidad de problemas. Los sistemas de ductos reconfigurables permiten formar una gran variedad de topologías de interconexión que acelera la ejecución de algoritmos. Para estos modelos existen bastantes algoritmos que corren en tiempo constante. En esta tesis se describe una simulación de una Rejilla Reconfigurable Dirigida (DR-Mesh) sobre una Rejilla Reconfigurable Lineal(LR-Mesh). La simulación entre modelos permite trasladar todos los algoritmos que se han diseñado para el modelo simulado, al modelo simulador. Las simulaciones actuales del DR-Mesh sobre el LR-Mesh requieren una gran cantidad de recursos. La simulación existente más rápida de un DR-Mesh de $N \times N$ procesadores sobre un LR-Mesh corre en $O(\log N)$ unidades de tiempo (u. t.) y utiliza $O(N^{12})$ procesadores en el LR-Mesh. La simulación existente más eficiente en recursos requiere $O\left(\frac{N^2}{\log N} \times \frac{N^2}{\log N}\right)$ procesadores en el LR-Mesh y tarda $O(\log^2 N)$ u.t. El objetivo de este trabajo es diseñar una simulación de un ciclo arbitrario de un DR-Mesh acíclico de $N \times N$ procesadores sobre un LR-Mesh. El tiempo de ejecución de esta simulación debe ser lo más cercano posible a $O(\log N)$ utilizando un número de procesadores reducido. La importancia del DR-Mesh está en su habilidad de ejecutar ciertas clases de algoritmos (sobre todo los relacionados con los grafos dirigidos) más rápido que los modelos con ductos no dirigidos. Por otro lado el LR-Mesh es un modelo más viable de ser elaborado que el DR-Mesh ya que sólo utiliza conexiones lineales y sus ductos no son dirigidos, estas características permiten que el modelo simulador pueda utilizar ductos ópticos sin ningún problema. El tiempo de ejecución de la simulación propuesta es $O(N^2)$, aunque aún no se ha determinado formalmente el tiempo de ejecución promedio, los resultados obtenidos en este trabajo indican que este tiempo es menor a N.

Palabras clave: Modelos paralelos, Algoritmos aleatorios, Mesh reconfigurable.

ABSTRACT of the thesis of JOSÉ ANTONIO CÁRDENAS HARO, submitted in partial fulfillment of the requirements to obtain the degree of MASTER in COMPUTER SCIENCE. Ensenada, Baja California, México. November 2001.

EFFICIENT SIMULATION OF PARALLEL MODELS DR-MESH ON LR-MESH

Abstract approved by:

Dr. José Alberto Fernández Zepeda
Thesis Director

This document examines the computational characteristics of some parallel models with reconfigurable buses. They have shown to be computationally more powerful than conventional models as the PRAM. The power of these models is given by their ability to dynamically alter the connections among processors. Reconfigurable buses permit to form a great variety of interconnection topologies that accelerates the performance of algorithms. Fast algorithms have been developed for these models to solve a large quantity of problems, many of them run in constant time. This thesis describes a simulation of a Directed Reconfigurable Mesh (DR-Mesh) on a Linear Reconfigurable Mesh (LR-Mesh). The simulation among models permits to transfer all the algorithms that have been designed for the simulated model, to the simulating model. Existing simulations of the DR-Mesh on the LR-Mesh require a large quantity of resources. The fastest simulation of a DR-Mesh of $N \times N$ processors on an LR-Mesh runs in $O(\log N)$ units of time (u.t.) and utilizes $O(N^{12})$ processors. The most efficient simulation requires $O\left(\frac{N^2}{\log N} \times \frac{N^2}{\log N}\right)$ processors in the LR-Mesh and takes $O(\log^2 N)$ u.t. The objective of this work is to design a simulation of an arbitrary cycle of an acyclic DR-Mesh of $N \times N$ processors on an LR-Mesh with optimal number of processors. The importance of the DR-Mesh stems from its ability to execute certain classes of algorithms (mostly those on directed graphs) faster than undirected models. Also, the construction of an LR-Mesh is more feasible than the DR-Mesh, since it utilizes only linear connections and its buses are undirected. These characteristics permit the simulating model to utilize optical buses without problems. The execution time of the proposed simulation is $O(N^2)$, although the average time has not been formally determined, our results indicate that this time is smaller than N.

Keywords: Parallel models, Randomized algorithms, Reconfigurable Mesh.

DEDICATORIA

A mis padres y hermanos por su siempre incondicional apoyo.

A *Beatriz* por su gran cariño y paciencia hacia mí, y porque por ella aprendí a percibir la armonía de la felicidad que encierra el verdadero amor.

A mi sobrina por alegrar mis días aún en su ausencia.

A *Linus Torvalds* por ser todo un revolucionario y por haberme ahorrado tantos dolores de cabeza.

A todos aquéllos que día a día luchan buscando hacer de este un mejor mundo para vivir.

AGRADECIMIENTOS

Al *Dr. José Alberto Fernández Zepeda* por proponer abordar el interesante tema y aceptar ser mi director de tesis en el mismo; así como por su valiosa ayuda.

A los miembros de mi comité de tesis, *M.C. José Luis Briseño, Dr. Andrei Tchernykh* y *Dr. José Frez Cárdenas* por sus comentarios, apoyo y tiempo.

Al personal del departamento de Ciencias de Computación, y a sus secretarias *Lydia* y *Caro*, que muy eficiente y amablemente atienden los constantes requerimientos de los estudiantes.

A todos los amigos de maestría que con su compañía y apoyo hicieron más amena mi estancia en este centro de investigación.

Al Centro de Investigación Científica y de Educación Superior de Ensenada.

Al Consejo Nacional de Ciencia y Tecnología por que sin su apoyo económico no hubiese sido posible la realización de esta maestría.

Índice General

Índice de Figuras

FIGURA PÁGINA

FIGURA PÁGINA

Capítulo I

Introducción

En la computación secuencial existe un modelo ampliamente aceptado, la máquina de acceso aleatorio (RAM), que se basa en la arquitectura de *von-Neumann*; en contraste, aún no existe el equivalente popular para la computación paralela. En particular, no es claro cuál modelo paralelo de computación es el mejor candidato para hacer la fusión "hardware-software". El modelo de máquina paralela de acceso aleatorio (PRAM) [JáJá, 1992; Sidhu, 1997] usualmente se considera como el ambiente computacional ideal por sus nulas restricciones en los accesos a memoria. Por otro lado, el modelo FCN (Red de Conexiones Fijas, por sus siglas en inglés), se considera como un ambiente paralelo "realizable", ya que cada elemento de procesamiento esta conectado a un número constante de otros elementos. El cómputo reconfigurable ha ganado una gran atención con la promesa de proporcionar un mejor desempeño que otros modelos paralelos. Existen varias líneas de investigación enfocadas a conocer el potencial del cómputo reconfigurable. Los modelos de ductos reconfigurables son en general más veloces que algunos modelos paralelos más populares como la PRAM.

I.1 Modelos de ductos reconfigurables

Algunos desarrollos recientes en la tecnología han hecho viables a los modelos de ductos reconfigurables, los cuales son más poderosos computacionalmente que la

PRAM. Como ejemplo de estos modelos se citan al Torus Polimórfico [Li y Maresca, 1989], rejilla reconfigurable (R-Mesh) [Miller *et al.*, 1993], arreglo de procesadores con sistema de ducto reconfigurable (PARBS) [Wang y Chen, 1990], máquina de ducto múltiple reconfigurable (RMBM) [Thiruchelvan *et al.*, 1993], entre otros. De éstos, la rejilla reconfigurable y sus variantes son los más estudiados.

Los modelos de ductos reconfigurables, son modelos de cómputo paralelo cuya característica principal está en la forma en como se manejan los ductos para la comunicación, posibilitando modelos de conexión flexibles mediante la conexión o desconexión de sus ductos. El objetivo básico de los ductos reconfigurables es flexibilizar la interconexión entre procesadores, mediante la conexión y desconexión de los puertos de cada procesador. Esto permite una gran variedad de topologías en la malla que aceleran la ejecución de algoritmos [Ben-Asher *et al.*, 1994]. La malla opera en ciclos bien definidos, y en cada uno de ellos la malla se reconfigura dinámicamente. Una arquitectura de ducto reconfigurable consiste de un arreglo de procesadores conectados entre sí a través de un número fijo de puertos de Entrada/Salida (E/S). Esta arquitectura es capaz de configurar una topología que contribuya a resolver el problema en curso. La estructura de un ducto se obtiene configurando localmente los interruptores dentro de cada procesador. Se pueden formar ductos de diferente forma como renglones, columnas, diagonal y zigzag, entre otros, usando los interruptores locales en los puertos de cada procesador.

En un ciclo de máquina, cada procesador realiza las siguientes funciones:

1. Configura un patrón de conexiones en sus puertos.

2. Escribe en los ductos conectados a sus puertos.

3. Lee de los ductos conectados a sus puertos.

4. Ejecuta operaciones aritméticas o lógicas usando los datos locales.

Las conexiones entre procesadores son fijas; mediante éstas y las conexiones internas entre puertos se forman los ductos. Un punto clave es que el proceso de reconfiguración de los ductos se lleva a cabo localmente en cada procesador de la malla. Esto es, al inicio de cada ciclo durante la ejecución de un programa en la malla, cada procesador fija su configuración localmente particionando o reconectando sus puertos. Cualquier procesador conectado a un ducto, puede elegir escuchar cualquier mensaje que se transmite por dicho ducto y/o usarlo para escribir un dato en él. La suposición básica concerniente al comportamiento de los modelos reconfigurables es que, en cualquier configuración, el tiempo requerido para transmitir un dato a lo largo del ducto es constante, independientemente del largo del ducto; aunque teóricamente esto es falso, ya que la velocidad de las señales que transportan información está limitada por la velocidad de la luz, y se tiene cierto retardo en los interruptores por el tiempo de respuesta del material semiconductor del que están hechos éstos. Sin embargo, esta suposición es una buena aproximación cuando los ductos son pequeños.

Recientemente se han incluido nuevas tecnologías en los modelos reconfigurables, incluyendo dispositivos ópticos para la comunicación y cómputo. La tecnología de muy alta escala de integración (VLSI) ofrece un buen ambiente para la construcción de sistemas de procesamiento paralelo con cientos de procesadores. Un esquema de interconexión muy atractivo es el de malla interconectada en dos dimensiones, por su simplicidad, regularidad y por el hecho de que el alambrado de interconexión ocupa únicamente una fracción fija del área sin importar el tamaño de la malla. Actualmente ya han sido construidas varias máquinas de reconfiguración dinámica con miles de interruptores, mostrándose así que este tipo de arquitecturas paralelas se pueden construir [Bondalapati y Prasanna, 1997].

Los sistemas de ductos reconfigurables permiten formar una gran variedad de topologías de interconexión [Moreira, 1997; Miller y Prasanna-Kumar,1993; Miller *et*

al., 1988b; Miller y Stout, 1996] que aceleran la ejecución de algoritmos [Bondalapati y Prasanna, 1997]. Para estos modelos existen bastantes algoritmos que corren en tiempo constante [Jang *et al.*, 1997] y se sugiere al parámetro *usanza de los ductos* (bus-usage) como una medida de la eficiencia en los algoritmos para modelos reconfigurables [Ben-Asher y Schuster, 1991; Schuster, 1991]; a mayor uso de la propiedad de reconfiguración de los ductos, se dice que el algoritmo es más eficiente. Por estos resultados y el creciente volumen de algoritmos se hace necesario un planteamiento más sistemático y una evaluación teórica de las clases de problemas que pueden resolverse utilizando modelos con ductos reconfigurables. En particular, es evidente que estos modelos resuelven una gran cantidad de problemas en un tiempo constante [Jang *et al.*, 1997; Stout, 1992]; esta capacidad se atribuye al gran número de configuraciones globales posibles en la malla en cada ciclo y la rapidez de propagación en los ductos.

Cuando el problema puede resolverse reconfigurando localmente de acuerdo a la entrada, entonces una configuración global da el resultado instantáneamente [Kapoor, 1993]. Se han propuesto muchos algoritmos eficientes para estos modelos. Algunos ejemplos son los algoritmos para grafos [Wang y Chen, 1990], algoritmos aritméticos [Nakano, 1994; Jang y Prasanna, 1997], de procesamiento de imágenes [Chung, 1996; Miller *et al.*, 1988], de geometría computacional [ElGindy y Wetherall, 1997] y de ordenamiento [Jang y Prasanna, 1995].

I.2 Simulación entre modelos

La simulación entre modelos permite trasladar todos los algoritmos que se han diseñado para el modelo simulado, al modelo simulador. Sean A y B dos modelos distintos de cómputo paralelo, el modelo B simula al modelo A si el modelo B puede "reproducir" el comportamiento de un paso arbitrario del modelo A. Esta simulación permite trasladar todos los algoritmos que se han diseñando para el modelo A al modelo

B. El costo de esta transformación es igual al costo de la simulación entre los modelos. Además da a conocer que tan poderoso computacionalmente es un modelo con respecto al otro.

I.3 Antecedentes

Las simulaciones actuales del DR-Mesh sobre el LR-Mesh requieren una gran cantidad de recursos. La simulación existente más rápida de un DR-Mesh de $N \times N$ procesadores sobre un LR-Mesh corre en $O(\log N)$ unidades de tiempo (u. t.) y se utilizan $O(N^{12})$ procesadores en el LR-Mesh [Ben-Asher *et al.*, 1991, Trahan *et al.*, 1998]. La simulación existente más eficiente en recursos requiere $O\left(\frac{N^2}{\log N} \times \frac{N^2}{\log N}\right)$ procesadores en el LR-Mesh y tarda $O(\log^2 N)$ u.t. [Fernández-Zepeda, 1999]. En el capítulo IV se hace una descripción más detallada de estas simulaciones.

I.4 Objetivos

El objetivo de este trabajo es diseñar una simulación de una rejilla reconfigurable dirijida (DR-Mesh) sobre una rejilla reconfigurable lineal (LR-Mesh); específicamente, se busca simular un ciclo de máquina arbitrario de un DR-Mesh de $N \times N$ procesadores sobre un LR-Mesh. Se busca que el tiempo de ejecución de esta simulación sea lo más cercano a $O(\log N)$, utilizando un número de procesadores reducido, específicamente menor al número de procesadores que emplean las simulaciones que se describen en el capítulo IV. Con la realización de este trabajo, también se busca comprender mejor la relación que existe entre diversos modelos reconfigurables y diseñar técnicas que agilicen la ejecución de algoritmos en estos modelos. La importancia del DR-Mesh está en su habilidad de ejecutar ciertas clases de algoritmos (sobre todo los relacionados con los grafos dirigidos) más rápido que los modelos con ductos no dirigidos. Por otro lado, el LR-Mesh es un modelo más viable de ser elaborado que el DR-Mesh ya que sólo utiliza

conexiones lineales y sus ductos no son dirigidos; estas características permiten que el modelo simulador pueda utilizar ductos ópticos sin ningún problema.

Para simular direccionalidad, un modelo con ductos no dirigidos puede particionar sus ductos para controlar la propagación de datos en segmentos específicos, pero mover los datos de un segmento a otro ocasiona un incremento en tiempo. Por ello no es factible que un R-Mesh simule a un DR-Mesh en tiempo constante, incluso con un incremento polinomial en el número de procesadores.

En el presente trabajo se propone una simulación de un DR-Mesh acíclico de $N \times N$ procesadores sobre un LR-Mesh de $O(N \times N)$, lo cual lo hace óptimo en recursos. Aunque no se ha determinado su tiempo de ejecución formalmente, los resultados muestran que la simulación propuesta es bastante rápida. A pesar de que el DR-Mesh se considera más poderoso computacionalmente que el LR-Mesh, en teoría es más sencillo y económico construir un LR-Mesh, por ello la importancia de esta simulación.

I.5 Organización

En el capítulo III se presentan algunos algoritmos básicos para modelos con ductos reconfigurables y se explica por que estos algoritmos son eficientes en este tipo de modelos. En el capítulo II se hace una descripción de los modelos utilizados en simulaciones previas y de los modelos que se utilizan en esta simulación. En el capítulo IV se describe y analiza el trabajo e investigaciones previas relacionados con esta investigación. El capítulo V se explican las etapas paso a paso de la simulación propuesta y su comprobación. En el capítulo VI se hace una evaluación del tiempo de ejecución promedio del algoritmo. Finalmente, en el capítulo VII se presentan las conclusiones de esta investigación y el trabajo futuro.

Capítulo II

Modelos utilizados

En esta sección se describen los modelos DR-Mesh, R-Mesh y LR-Mesh. Así como una introducción a los grafos, y como éstos se utilizan para representar el patrón de conexiones, y la transferencia de datos de los modelos con ductos reconfigurables. Se mencionan además los circuitos *fan-in* y la PRAM. Al igual que en la PRAM con lecturas y escrituras concurrentes (CRCW), existen ciertas reglas para resolver los problemas con las escrituras concurrentes en los modelos con ductos reconfigurables. En este caso, los problemas con escrituras concurrentes se resuelven mediante las reglas *Común, Colisión, Colisión*$^+$, *Prioridad,* o *Arbitrario.* La regla *Común* permite escrituras concurrentes únicamente si todos los valores que se escriben en el ducto son iguales. Bajo la regla de *Colisión*, si más de un procesador intenta escribir en un ducto, entonces se genera el símbolo de colisión. La regla de *Colisión*$^+$ se comporta como *Común* cuando todos los procesadores intentan escribir el mismo valor en el ducto, de otra manera se comporta como *Colisión.* Con la regla de *Prioridad*, el procesador con la más alta jerarquía, ya sea el que tenga el índice mayor o menor según sea el caso, es el que escribe en el ducto. Con la regla *Arbitrario* si más de un procesador intenta escribir en el ducto, cualquiera de ellos de manera aleatoria, es seleccionado para que lo haga.

II.1 Rejilla reconfigurable (R-Mesh)

Es un arreglo bidimensional de procesadores conectados en forma de cuadrícula. Cada procesador en el R-Mesh tiene conexiones fijas con los procesadores adyacentes a través de sus cuatro puertos (norte, sur, este, oeste) de entrada/salida. Cada procesador puede conectar y desconectar internamente sus puertos de acuerdo a decisiones locales. Esto permite al R-Mesh hacer quince diferentes configuraciones de ducto y cambiarlas dinámicamente de acuerdo a los requerimientos del problema a resolver. El R-Mesh asume un tiempo de propagación constante en los ductos [Li y Stout, 1991; Miller *et al.*, 1993], independientemente del número de puertos conectados a él. La figura 1 muestra un R-Mesh de 3×5 procesadores con sus quince diferentes configuraciones internas de ductos, una en cada procesador; la linea más gruesa muestra un componente conectado.

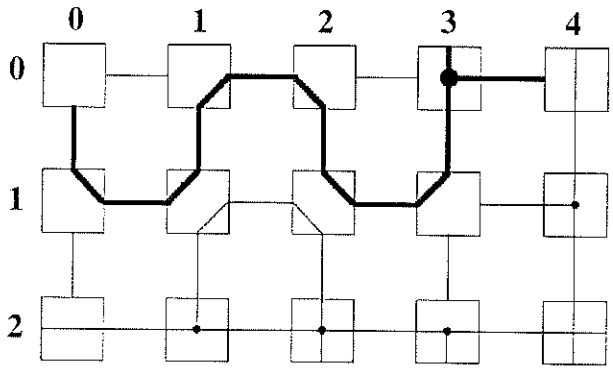


Figura 1: R-Mesh de 3×5.

II.2 Rejilla reconfigurable lineal (LR-Mesh)

Esta es una versión restringida del R-Mesh. Cada puerto en un LR-Mesh puede conectarse a lo más con otro puerto dentro del mismo procesador; de esta forma el LR-Mesh permite solo diez de las quince conexiones del R-Mesh. Los ductos construidos

en este modelo se denominan lineales. La figura 2 muestra un LR-Mesh de 3×5 procesadores.

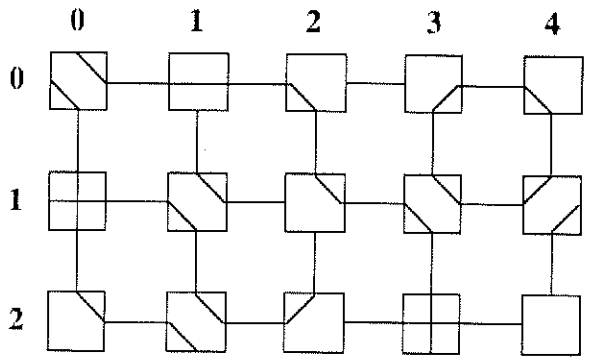


Figura 2: LR-Mesh de 3×5.

II.3 Rejilla reconfigurable dirigida (DR-Mesh)

La estructura de un DR-Mesh difiere de la de un R-Mesh en que el DR-Mesh posee ductos dirigidos mientras que el R-Mesh utiliza ductos no dirigidos. Los datos en un ducto dirigido se propagan solamente en una dirección. Cada procesador en el DR-Mesh tiene cuatro puertos de salida conectados a ductos de salida y cuatro puertos de entrada conectados a ductos de entrada. Esto permite 4140 conexiones distintas entre los puertos de un procesador. La figura 3 muestra un DR-Mesh de 3×5 procesadores.

II.4 Relación entre modelos reconfigurables

La clase de lenguajes aceptados por un LR-Mesh (R-Mesh, DR-Mesh, respectivamente) en un tiempo constante con un número polinomial de procesadores, es equivalentes a la clase L (SL, NL, respectivamente) de lenguajes aceptados con un espacio logarítmico determinístico (espacio logarítmico simétrico, espacio logarítmico no determinístico, respectivamente) en una máquina de Turin. Una conjetura ampliamente

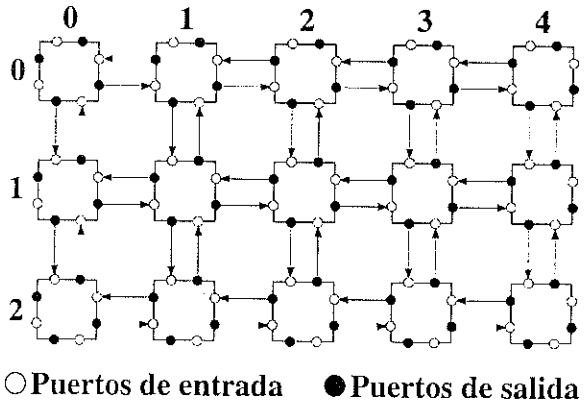


Figura 3: DR-Mesh de 3×5.

aceptada es que $L \subset SL \subset NL$, por lo que es probable que el LR-Mesh sea un modelo computacionalmente más débil que el R-Mesh, y el R-Mesh a su vez más débil que el DR-Mesh [Ben-Asher *et al.*, 1995].

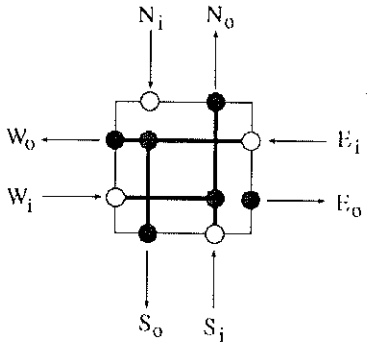
II.5 Grafos dirigidos

Muchos problemas en ingeniería y en ciencias se pueden formular en términos de grafos dirigidos. El diseño de algoritmos paralelos para resolver problemas de grafos dirigidos es por ello de interés teórico e importancia práctica. Una representación conveniente de un grafo $G = (V, E)$ es la matriz de adyacencias, se denota a esta matriz como A. El elemento (i,j) de A se representa por $A_{i,j}$. El elemento $A_{i,j} = 1$ si $(i, j) \in E$ o si $i = j$; $A_{i,j} = 0$ de cualquier otra manera, $1 \leq i, j \leq n$. El *cierre transitivo* (*transitive closure*) de G es un grafo $G^* = (V, E^*)$, donde la arista $\langle v_1, v_2 \rangle \in E^*$, si $v_1, v_2 \in V$ y además existe una trayectoria de v_1 a v_2. La matriz del *cierre transitivo* de G, denotada como A^*, se define como sigue: $A^*_{i,j} = 1$ si existe una ruta entre el nodo i y el nodo j o si $i = j$; $A^*_{i,j} = 0$ de otra manera. Si G es dirigido, A y A^* no son necesariamente simétricas. A^* se llama también matriz de conectividad. Mediante el

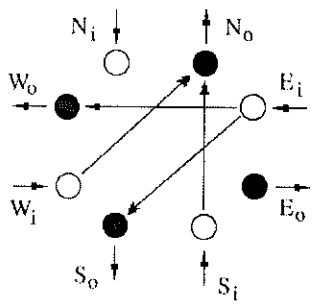
cierre transitivo se puede saber hasta donde llega un dato que ha sido escrito en uno de los puertos del modelo dirigido y emularlo así en el modelo simulador.

II.6 Grafo equivalente de un DR-Mesh

El patrón de conexiones de un DR-Mesh se puede representar como un grafo dirigido, donde los nodos representan a los puertos del procesador. Como se puede observar en la figura 4, una arista dirigida conecta a dos nodos en el grafo, únicamente si en el procesador del DR-Mesh hay una conexión entre dichos nodos. La matriz de adyacencias es una representación del grafo, en donde cada nodo corresponde a un renglón y/o columna. La figura 4a muestra las particiones de puertos en un procesador de un DR-Mesh, la figura 4b muestra el grafo dirigido correspondiente, y la figura 4c, la matriz de adyacencias.



(a)



(b)

	N_i	N_o	E_i	E_o	W_i	W_o	S_i	S_o
N_i	1	0	0	0	0	0	0	0
N_o	0	1	0	0	0	0	0	0
E_i	0	0	1	0	0	1	0	1
E_o	0	0	0	1	0	0	0	0
W_i	0	1	0	1	1	0	0	0
W_o	0	0	0	0	0	1	0	0
S_i	0	1	0	0	0	0	1	0
S_o	0	0	0	0	0	0	0	1

(c)

Figura 4: a) Ejemplo de configuración de puertos en un procesador de DR-Mesh; b) Grafo equivalente de los puertos; c) Matriz de adyacencias del grafo.

En la figura 5 se ilustra un grafo con su *cierre transitivo* y sus respectivas matrices de adyacencia.

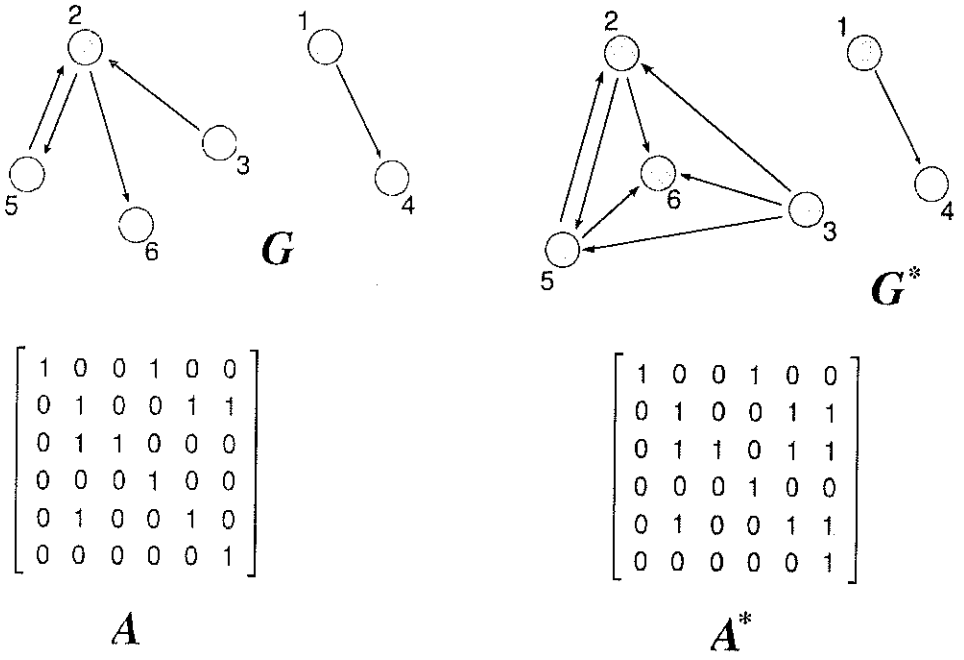


Figura 5: Ejemplo de un grafo G y su *cierre transitivo* G^*. A y A^* son las matrices de adyacencia de G y G^* respectivamente. Las entradas sombreadas que aparecen en A^* indican las aristas que están en G^* pero no en G. Los autolazos de cada nodo en G y G^* se omiten por simplicidad, aunque $a_{ii} = a^*_{ii} = 1$ para cada i.

II.7 Circuitos Fan-in

Un circuito se puede ver como un grafo acíclico dirigido, en donde los nodos con grado de entrada cero son los datos de entrada. Todos los demás nodos tienen un grado de entrada de al menos dos y representan operadores lógicos ya sean AND u OR. Los nodos con grado de salida cero son las salidas del circuito [Gál, 1995; Trahan *et al.*, 1998]. Un circuito tiene *Fan-in* acotado si el *Fan-in* de todas las compuertas está restringido a una constante; *Fan-in* no acotado, si el *Fan-in* no tiene restricciones; y *Fan-in* semiacotado, si el *Fan-in* de las compuertas OR no tiene restricciones, mientras que el *Fan-in* de las compuertas AND está restringido a una constante. El tamaño del circuito está dado por la cantidad de conexiones entre compuertas que posee y la

profundidad es la ruta más larga que existe desde cualquier entrada a cualquier salida del circuito.

II.8 Máquina paralela de acceso aleatorio

Este modelo es una extensión del modelo básico secuencial. En la máquina paralela de acceso aleatorio (PRAM) [JáJá, 1992], todos los procesadores tienen acceso a una unidad única de memoria compartida. Este modelo consiste de un conjunto de procesadores donde cada uno tiene su propia memoria local y pueden ejecutar sus programas localmente. Todos ellos se comunican intercambiando datos a través de la unidad de memoria compartida. Cada procesador se identifica por un índice. La figura 6 muestra al modelo PRAM con p procesadores. Estos procesadores tienen los índices 1, 2, ...,p. La memoria compartida también se llama memoria global.

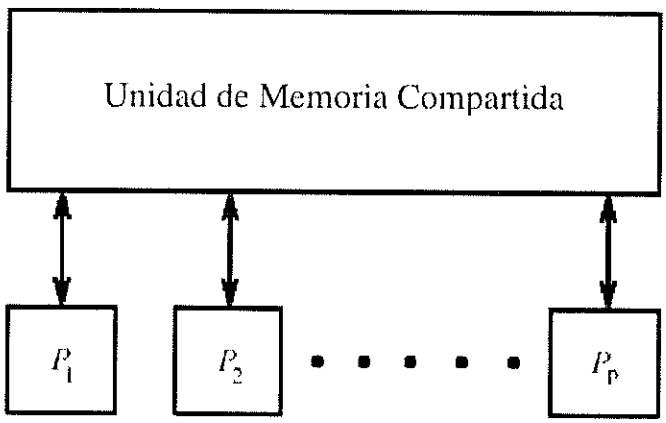


Figura 6: Modelo PRAM.

Hay dos modos básicos de operación del modelo PRAM. En el primer modo, llamado síncrono, todos los procesadores operan síncronamente bajo el control de un reloj común. En el segundo modo, llamado asíncrono, cada procesador opera con su propio reloj. En el modo asíncrono de operación, es responsabilidad del programador ajustar los puntos de sincronización apropiados cada vez que sea necesario. Si un

procesador requiere acceder a cierto dato, el programador debe asegurar que se obtengan los valores correctos, ya que el valor de una variable compartida se determina dinámicamente durante la ejecución de los programas en los diferentes procesadores. Como cada procesador puede ejecutar su propio programa local, el modelo PRAM es del tipo de instrucciones múltiples y datos múltiples, MIMD por sus siglas en inglés.

Capítulo III

Cómputo paralelo en rejillas reconfigurables

La tecnología de integración a gran escala (VLSI) ofrece un buen ambiente para la construcción de sistemas de procesamiento paralelo que consisten de miles de procesadores. Un esquema muy atractivo de interconexión es el de malla bidimensional, por su simplicidad, regularidad, y por el hecho de que las lineas de interconexión entre procesadores ocupan únicamente un valor fraccionario constante del área, independientemente del tamaño de la malla. Dado que una malla de tamaño N^2 esta configurada como un arreglo de $N \times N$ procesadores, el diámetro de comunicación (el máximo de la distancia mínima entre dos procesadores cualquiera) es $O(N)$. En una malla con ductos reconfigurables el tiempo de comunicación entre dos procesadores cualquiera es $O(1)$.

Muchos algoritmos se han diseñado para correr en un tiempo $O(1)$ en mallas con ductos reconfigurables, éstos continuamente reconfiguran a todo el sistema cambiando la interconexión de los ductos mediante la manipulación de los interruptores en los procesadores, obteniendose así los patrones de conexión deseados. De ello, se han derivado algoritmos que se benefician al máximo de la transmisión de datos en renglones y columnas [Bondalapati y Prasanna, 1997; Ben-Asher *et al.*, 1991; Jang *et al.*, 1997; Nakano, 1994; Stout, 1992].

Se han diseñado algoritmos para el computo básico como es el cálculo del OR, AND, OR exclusiva (OR-EX), adición, multiplicación, máximos o mínimos, etc.; siendo éstos óptimos para las diferentes variantes de modelos con ductos reconfigurables. Utilizando las operaciones básicas mencionadas, y técnicas adicionales no triviales para explotar al máximo la reconfiguración en los ductos, se obtienen algoritmos para resolver problemas de procesamiento de imágenes, de geometría computacional, grafos, etc.

A continuación, se describen algunos algoritmos desarrollados por otros autores que han sido optimizados para modelos con ductos reconfigurables. Estos algoritmos ilustran el poder de estas arquitecturas, como es el caso del algoritmo para calcular a la OR-EX, o la suma, o el máximo o mínimo de N números, todos ellos en un tiempo constante.

III.1 Cálculo de la función OR-EX

El cálculo de la función OR-Exclusiva (OR-EX) de n bits se realiza en una malla reconfigurable de tamaño $2n \times 3$ en un tiempo $O(1)$ [Miller y Prasanna-Kumar,1993]. La idea básica del funcionamiento de este algoritmo se describe a continuación. Basados en un simple bit de entrada en un arreglo de 3×2 procesadores, cada arreglo configura sus puertos con uno de los dos patrones mostrados en la figura 7. Si el bit de entrada es 0, la configuración adoptada en este caso es como la que se muestra en la figura 7a, donde todos los procesadores conectan sus puertos este con oeste únicamente, obteniendo así un arreglo horizontal de los ductos. De esta forma, un dato que ingrese por el puerto oeste del procesador $P_{0,2j}$ pasa directamente al procesador $P_{0,2j+1}$ y sale por su puerto este. De la misma forma se transfiere un dato para el segundo renglón.

Si el bit de entrada es 1, la configuración adoptada es como la que se muestra en la figura 7b; así, si el dato entra por el puerto oeste del procesador $P_{0,2j}$, éste se transfiere

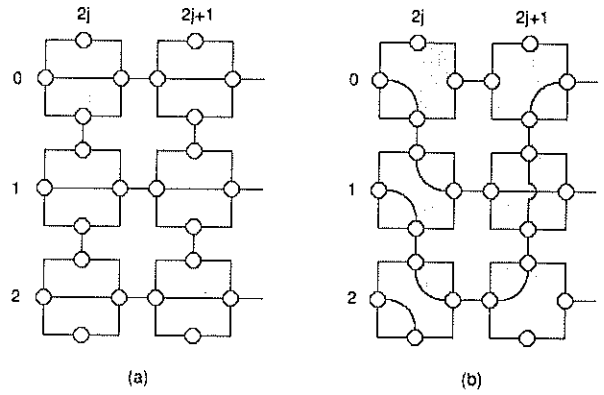


Figura 7: Conexiones para el cálculo de la función OR-EX; a) para entrada 0; b) para entrada 1.

al segundo renglón y sale por el puerto este del procesador $P_{1,2j+1}$, o si el dato entra por el puerto oeste del procesador $P_{1,2j}$, éste es transferido al primer renglón y sale por el puerto este del procesador $P_{0,2j+1}$.

Una señal de entrada se aplica al puerto oeste del procesador $P_{0,0}$ de la malla formada, y se transfiere a lo largo de la malla atravesando todas las columnas. Si en la última columna la señal sale por el primer renglón, indica un resultado de 0; si la señal sale por el segundo renglón de la última columna, ésto indica un resultado de 1.

Un ejemplo del cálculo de la OR-EX para tres bits de entrada se muestra en la figura 8, en este caso se requiere una malla de 3×6 procesadores. Las lineas resaltadas muestran la trayectoria del dato de entrada que ingresa por el puerto oeste del procesador $P_{0,0}$, y sale por el puerto este del procesador $P_{0,5}$. La malla realiza la operación OR-EX de las entradas 1, 0, 1, su resultado igual a 0.

III.2 Sumas

La suma de dos números de n bits puede realizarse en forma similar a como se realiza el cálculo de la OR-EX. En este caso cada procesador toma dos bits, uno de cada número, de la misma posición significativa; la malla requerida es de tamaño $1 \times n$.

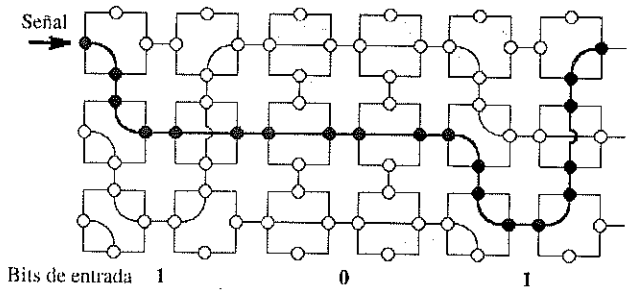


Figura 8: Muestra el cálculo de $1 \oplus 0 \oplus 1 = 0$.

Si el procesador va a sumar dos bits diferentes, entonces adopta un sus puertos una configuración de *propagador de acarreo*, y conecta su puerto oeste con su puerto este. Si el procesador va a sumar dos unos, la configuración adoptada es de *generador de acarreo*, donde no hay conexión interna en los puertos pero escribe un uno en su puerto oeste. En el caso de que el procesador va a sumar dos ceros, la configuración que toma es la de *anulador de acarreo*, donde no hay conexión entre los puertos internos ni escritura en los puertos.

Un ejemplo de adición de dos números de cinco bits (a y b) se muestra en la figura 9. Los procesadores 0, 1 y 3, adoptan una configuración de propagación de acarreo, el procesador 2 toma una configuración de generador de acarreo, y por último el procesador 4 se configura como anulador de acarreo. Los bits c_i indican el acarreo intermedio y z_i son los bits resultantes que se obtienen de la operación $a_i \oplus b_i \oplus c_i$, y son la suma de los números binarios a y b.

Usando una idea similar y construyendo un sumador de acarreos en un lazo de k etapas, la adición de n números de k bits donde $1 \leq k \leq n$ se efectúa en un tiempo constante usando un modelo de malla con ductos reconfigurables de tamaño $n \times nk$ [Jang y Prasanna, 1995].

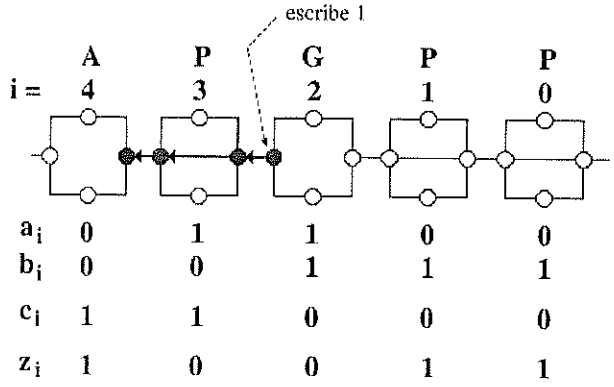


Figura 9: Adición de dos números de 5 bits.

III.3 Componentes conectados

Un grafo no dirigido está conectado si hay una ruta que conecta a cualquier par de sus nodos. Un grafo que no está conectado puede dividirse en subgrafos o componentes conectados. Con el algoritmo de componentes conectados se determinan estos componentes. En una malla con ductos reconfigurables, esto se hace en un tiempo constante.

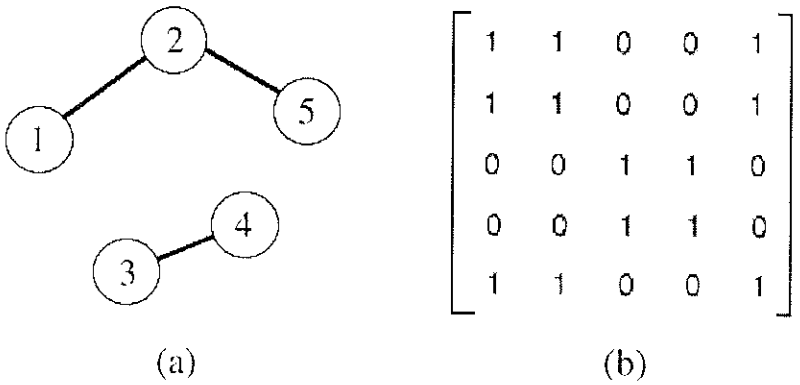


Figura 10: a) grafo; b) matriz de adyacencias.

Encontrar los componentes conectados constituye la base de muchas aplicaciones para grafos. Por ejemplo, considérese el problema de identificar cúmulos de elementos con las mismas características dentro de un gran bloque. Se puede representar a cada

elemento como un nodo y agregar una arista entre cada par de elementos similares. Entonces, los componentes conectados de este grafo corresponden a diferentes clases de elementos. Probar que elementos del grafo están conectados con cuales es un paso esencial de preprocesado para muchos algoritmos de grafos.

Para calcular los componentes conectados se requiere a la matriz de adyacencias. La figura 10 muestra un grafo con dos componentes y su matriz de adyacencias. El tamaño de la matriz es de 5×5, ya que el grafo consta de 5 nodos. Para cada nodo corresponde un renglón y una columna; si dos nodos son vecinos, el elemento correspondiente en la matriz vale uno, de lo contrario su valor es cero, por ello la matriz resultante es simétrica. Se considera que cada nodo tiene una trayectoria hacia si mismo; por ello, la diagonal principal es uno.

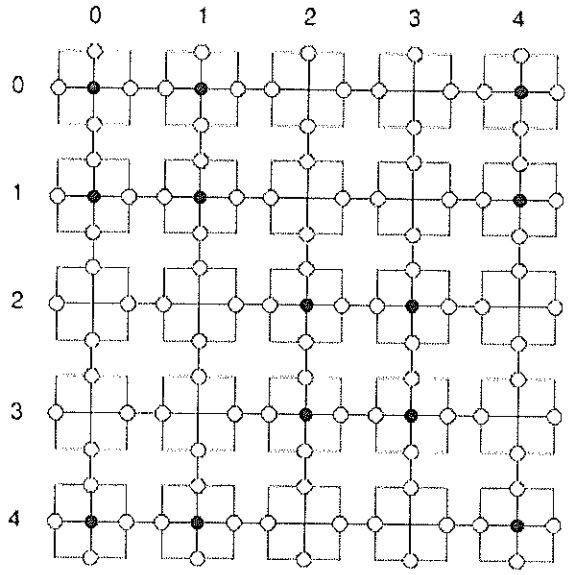


Figura 11: R-Mesh 5×5 para calcular componentes conectados.

El cálculo de los componentes conectados usando una malla con ductos reconfigurables es sencillo y se logra en un tiempo $O(1)$. Se requiere una malla de las mismas dimensiones que la matriz, $N \times N$, donde N es el número de nodos en el grafo. Cada

procesador almacena un elemento de la matriz de adyacencias. Cada procesador une sus cuatro puertos si el elemento correspondiente en la matriz es igual a uno; de lo contrario, sólo conecta su puerto norte con su puerto sur, y su puerto oeste con su puerto este, como se aprecia en la figura 11.

Todos los procesadores del renglón superior escriben el índice de su columna en su ducto vertical y después leen del mismo ducto. Se asume que el R-Mesh usa la regla de prioridad, por lo que los procesadores $P_{0,0}, P_{0,1}$ y $P_{0,4}$ leen el índice 0 (escrito por el procesador $P_{0,0}$) y así saben que pertenecen al componente conectado "0". De igual forma los procesadores $P_{0,2}$ y $P_{0,3}$ leen el índice "2" (escrito por el procesador $P_{0,2}$) que corresponde al componente conectado 2.

Capítulo IV

Trabajo previo

Existen tres simulaciones de un DR-Mesh sobre un LR-Mesh diseñadas por otros autores. La primera de ellas [Ben-Asher *et al.*, 1991; Trahan *et al.*, 1998] es indirecta y, aunque eficiente en tiempo ($O(\log N)$), requiere demasiados recursos ($O(N^{12})$ procesadores); la segunda simulación [Trahan *et al.*, 1997; Fernández-Zepeda *et al.*, 1999] corre en el mismo tiempo que la anterior y la cantidad de recursos que requiere es menor ($O(N^8)$); mientras que la tercera simulación [Fernández-Zepeda *et al.*, 1999], aunque ligeramente menos rápida ($O(\log^2 N)$), es más eficiente en recursos ($O(\frac{N^4}{\log^2 N})$). Estas simulaciones se describen a continuación.

IV.1 Primera simulación de un DR-Mesh sobre un LR-Mesh.

La primera simulación que se describe en esta sección es indirecta, y se lleva a cabo a través de un conjunto de modelos como se muestra en la figura 12.

Figura 12: Simulación indirecta de DR-Mesh en un LR-Mesh.

Esta simulación [Ben-Asher *et al.*, 1991; Trahan *et al.*, 1998] requiere de $O(\log N)$ pasos, lo cual la hace muy rápida aunque requiere de una gran cantidad de procesadores para efectuarla. Para realizar esta simulación se construye un grafo equivalente del DR-Mesh, en donde los puertos del DR-Mesh son los nodos y las conexiones entre ellos son las aristas (ver grafos en capítulo II). Después se calcula el *cierre transitivo* (*transitive closure*) de este grafo para determinar los destinos de los mensajes escritos en los puertos. Cabe señalar que el uso del *cierre transitivo* para simular un DR-Mesh es muy costoso en recursos, ya que un DR-Mesh de $N \times N$ tiene N^2 procesadores y 8 puertos por procesador; su representación como grafo requiere de $O(N^2)$ nodos y $O(N^2)$ aristas. La matriz de adyacencias que se obtiene a partir del grafo es de tamaño $O(N^2) \times O(N^2)$, lo cual equivale a $O(N^4)$ elementos. Para calcular el *cierre transitivo* se requiere multiplicar $O(\log N)$ veces por sí misma a la matriz de adyacencias [JáJá, 1992]. Durante cada multiplicación matricial, el cálculo de cada elemento de la nueva matriz requiere de $O(N^2)$ procesadores para ejecutarse en tiempo constante, lo que resulta en $O(N^6)$ procesadores.

Trahan [Trahan *et al.*, 1997] simuló al DR-Mesh en un circuito semi acotado (ver figura 4) utilizando el *cierre transitivo*. Ellos realizan la multiplicación de matrices antes mencionada usando circuitos lógicos (dado que las matrices son booleanas). El circuito lógico requerido consiste de $O(N^6 \log N)$ compuertas, la profundidad del circuito es $\log N$ (ver figura 13(b)). Cada uno de los bloques de la figura 13(b) consiste de $O(N^2)$ compuertas AND con dos entradas, cada una de las salidas de estas compuertas se conectan a la entrada de una compuerta OR (ver figura 13(a)).

Una vez que la simulación está hecha en el circuito semi acotado, ésta se traslada a un circuito no acotado. Esto se hace en un tiempo constante ya que el circuito semi acotado es un caso especial del no acotado. Después se simula al circuito no acotado en una PRAM CRCW [Karp y Ramachandran, 1990, Stockmeyer y Vishkin, 1984].

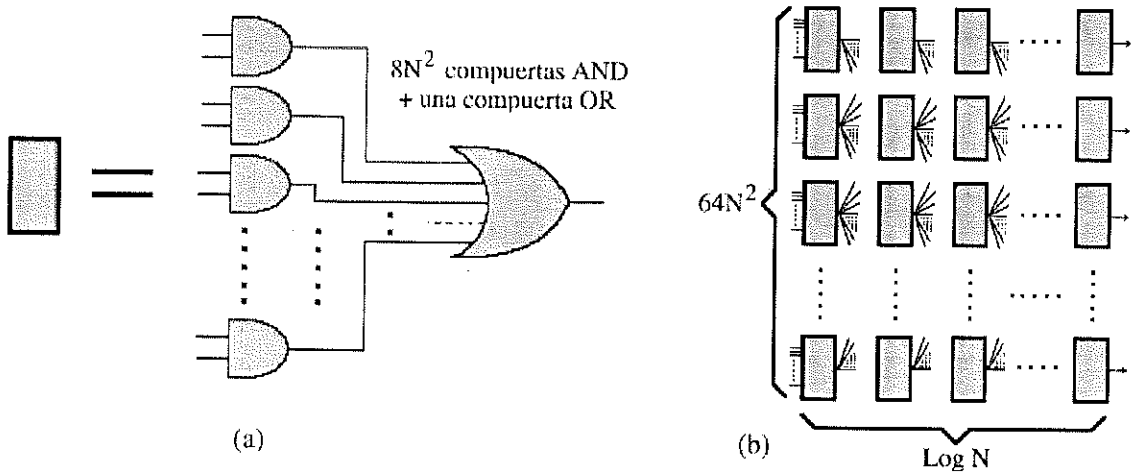


Figura 13: Circuito lógico empleado para realizar multiplicaciones de matrices booleanas; a) arreglo de compuertas lógicas equivalentes a un bloque; b) bloques de compuertas para resolver la matriz de adyacencias.

Cualquier circuito *fan-in* no acotado de n entradas, M compuertas, profundidad D y S aristas, se puede simular en una PRAM CRCW en un tiempo $O(D)$ con S procesadores y $n + M$ localidades de memoria [Karp y Ramachandran, 1990] (ver figura 14).

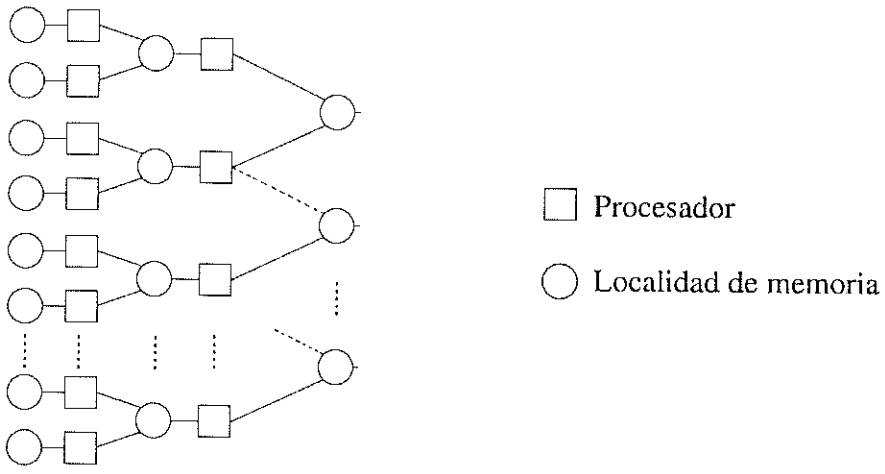


Figura 14: Simulación de circuitos lógicos en una PRAM.

En la simulación, se asigna un procesador a cada arista del circuito y una localidad de memoria para cada compuerta lógica. Inicialmente, las entradas están en las localidades de memoria desde 1 hasta n, y la localidad de memoria $n + i$ se asigna a la compuerta i en el circuito, $i = 1, \ldots, M$; se inicializan las salidas de las compuertas OR a 0 y las compuertas AND a 1.

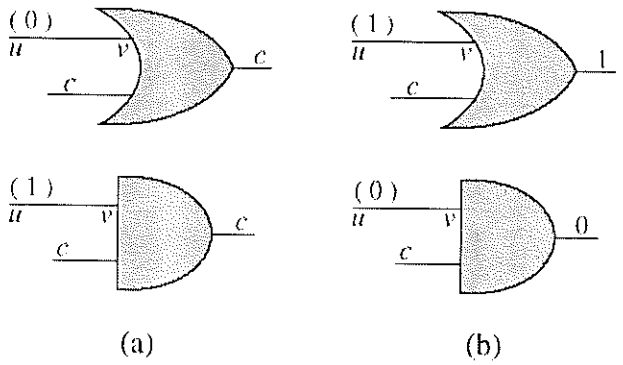


Figura 15: Consideraciones para la simulación de circuitos lógicos.

Cada paso en la simulación consiste de realizar las siguientes tres etapas en secuencia:

a) Cada procesador p determina el valor booleano de la arista que representa, $e = (u, v)$, leyendo la localidad de memoria $n + u$, que corresponde a la salida de la compuerta u.

b) Si el valor en su arista es 0 y v corresponde a una compuerta OR, o si el valor en su arista es 1 y v corresponde a una compuerta AND, entonces el procesador no hace nada ya que dichos valores de entrada no pueden alterar el valor de salida de la compuerta v (ver figura 15a).

c) Si el valor en su arista es 1 y v corresponde a una compuerta OR, o si el valor en su arista es 0 y v corresponde a una compuerta AND, entonces el procesador escribe

en la localidad de memoria $n + v$ el valor de dicha arista (ver figura 15b). La localidad de memoria $n + v$ representa la salida de la compuerta v.

Después de D pasos, la localidad de memoria $n + i$ tiene el valor de la compuerta i para $i = 1, \ldots, M$. Esto se debe a que se requieren D pasos para que se propague el efecto de todas las entradas a todas las salidas. El circuito lógico consta de $O(N^6)$ compuertas, $O(N^6)$ aristas, profundidad $\log N$ y $O(N^4)$ entradas (por lo tanto la PRAM requiere $O(N^6)$ procesadores y $O(N^6)$ localidades de memoria).

Por último Ben-Asher [Ben-Asher *et al.*, 1991] simula a la PRAM CRCW en el LR-Mesh. Ben-Asher demostró que una PRAM CRCW con regla de escritura de prioridad, con P procesadores y M celdas de memoria, puede simularse en tiempo constante en un LR-Mesh de tamaño $P \times M$. El tamaño del LR-Mesh es por lo tanto $O(N^6) \times O(N^6)$, que equivale a $O(N^{12})$ procesadores en total.

En la figura 16 se puede ver un esquema de esta simulación. Cada procesador en la columna de la izquierda (o columna 0) corresponde a un procesador de la PRAM CRCW. Los procesadores del renglón inferior se utilizan para emular a las localidades de memoria de la PRAM, el resto de los procesadores del LR-Mesh sólo se utilizan como interruptores y unidades de almacenamiento temporal.

A continuación se considera el caso de lectura en detalle; el de escritura, por ser un caso particular del de lectura, se omite. Para el caso de lectura, cada procesador $p_{i,0}$ (correspondientes a la primera columna) transmite sobre su renglón la dirección de la localidad de memoria j que quiere leer.

El interruptor de intersección (en este caso el procesador $p_{i,j}$ del LR-Mesh) localizado en el renglón i y la columna j, transmite la solicitud sobre su columna a los procesadores inferiores, al mismo tiempo que desconecta su puerto norte. En el siguiente paso, el procesador $p_{P-1,j}$ (localidad de memoria j) transmite, en respuesta, el

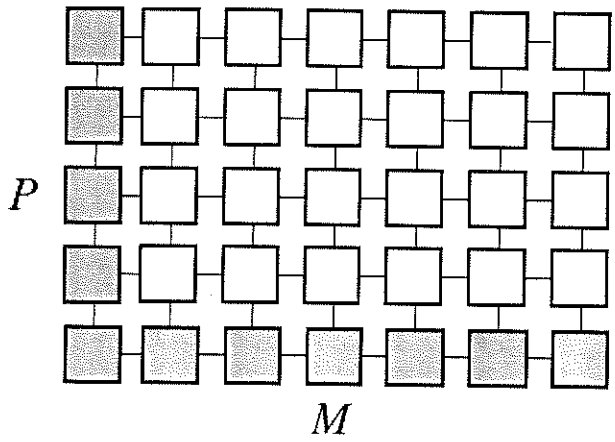


Figura 16: Simulación de una CRCW PRAM en un LR-Mesh.

elemento de memoria solicitado sobre su columna hacia todos los procesadores de la columna j, el procesador $p_{i,j}$ lo retransmite a través del ducto horizontal del renglón i hacia el procesador $p_{i,0}$. Para la etapa de escritura, se sigue un procedimiento similar al de la solicitud de lectura.

La desconexión del puerto norte resuelve los conflictos de lecturas múltiples al asignarse mayor prioridad a los procesadores de renglones inferiores para que hagan su solicitud. Así, sólamente una solicitud llega a la correspondiente localidad de memoria, aunque el contenido de memoria se envía a todos aquellos que pudieron haberlo solicitado.

Sea DR-Mesh^j (LR-Mesh^j, PRAM^j, respectivamente) la clase de problemas que se pueden resolver en un DR-Mesh (LR-Mesh, PRAM, respectivamente) en un tiempo $O(\log^j N)$ con un número polinomial en N de procesadores. Similarmente, para circuitos se utiliza AC^j (SAC^j, respectivamente) para indicar la clase de problemas que se pueden resolver con circuitos *fan-in* no acotados (semi acotados, respectivamente) de profundidad $O(\log^j N)$ y tamaño polinomial en N. Usando esta nomenclatura se puede establecer la siguiente relación para indicar la simulación indirecta del DR-Mesh en el LR-Mesh, pasando por circuitos y por una PRAM, y que se acaba de describir.

$$\text{DR-Mesh}^j \subseteq SAC^{j+1} \subseteq AC^{j+1} \subseteq PRAM^{j+1} \subseteq \text{LR-Mesh}^{j+1} \qquad \text{para } j \geq 0$$

Esta relación indica un incremento en tiempo de $O(\log N)$ al pasar del DR-Mesh al SAC. En las fases subsecuentes, el tiempo de la simulación se mantiene constante pero se requiere una gran cantidad de recursos como se aprecia en el análisis.

IV.2 Segunda simulación de un DR-Mesh sobre un LR-Mesh

Una segunda simulación también indirecta de un DR-Mesh sobre un LR-Mesh (ver figura 17) se basa en el trabajo de Trahan [Trahan *et al*, 1997] y Fernández-Zepeda [Fernández-Zepeda *et al*, 1999]. El primero demostró que cada paso de un DR-Mesh de $N \times N$ puede simularse en un R-Mesh de $O(N^4) \times O(N^4)$ en un tiempo $O(\log N)$ utilizando el *cierre transitivo*.

Figura 17: Simulación indirecta del DR-Mesh en el LR-Mesh.

Para realizar esta simulación, se construye un grafo donde los puertos de un DR-Mesh son los nodos; entonces, calculando el *cierre transitivo*, se determinan los destinos de los mensajes escritos en los puertos. A su vez, Fernández-Zepeda demostró que un R-Mesh de $N \times N$ se puede simular en un LR-Mesh de $N \times N$ en un tiempo $O(\log N)$. Por lo que esta simulación del DR-Mesh sobre el LR-Mesh requiere de un tiempo $O(\log^2 N)$.

Esta simulación no es tan rápida como la anterior y sigue requiriendo demasiados procesadores, esto es $O(N^8)$.

IV.3 Tercera simulación de un DR-Mesh sobre un LR-Mesh

La tercera simulación de un DR-Mesh sobre un LR-Mesh es una simulación directa diseñada por Fernández-Zepeda [Fernández-Zepeda *et al.*, 1999]. En su simulación, se resuelve el *cierre transitivo* en forma recursiva haciendola más eficiente. Esta simulación requiere $O(\log^2 N)$ pasos con $O\left(N \times N \times \frac{N}{\log N}\right)$ procesadores en un arreglo de tres dimensiones u $O\left(\frac{N^2}{\log N} \times \frac{N^2}{\log N}\right)$ procesadores en dos dimensiones. Esta simulación se explica brevemente a continuación.

Para hacer la explicación más sencilla, se describe primero la simulación en el modelo tridimensional, LR-Mesh CRCW, y se transforma después para simularse en el modelo bidimensional LR-Mesh CREW. Para simular un DR-Mesh de $N \times N$, el LR-Mesh simulador usa $16N \times 16N \times \frac{N}{\log N}$ procesadores. En las primeras dos dimensiones, un grupo de 16×16 procesadores del modelo simulador, se encargan de simular un sólo procesador del DR-Mesh. Los procesadores correspondientes a la tercera dimensión, incrementan la velocidad de la simulación, en operaciones tales como multiplicación de matrices y movimientos de datos. La simulación consta a grandes rasgos de dos fases, *Going_Out* y *Going_In*. El procedimiento *Going_Out* consiste en iterar desde pequeños bloques de procesadores (mosaicos), a bloques cada vez mayores hasta abarcar a toda la malla. Este algoritmo trabaja con el *cierre transitivo* y requiere de $\log N$ iteraciones. En la figura 18, se ejemplifica como se propagan los datos en el procedimiento de *Going_Out*.

La segunda fase de la simulación se llama *Going_In*; esta fase opera en forma inversa a la *Going_Out*. Aquí se utilizan los datos que salen de cada bloque de procesadores, y

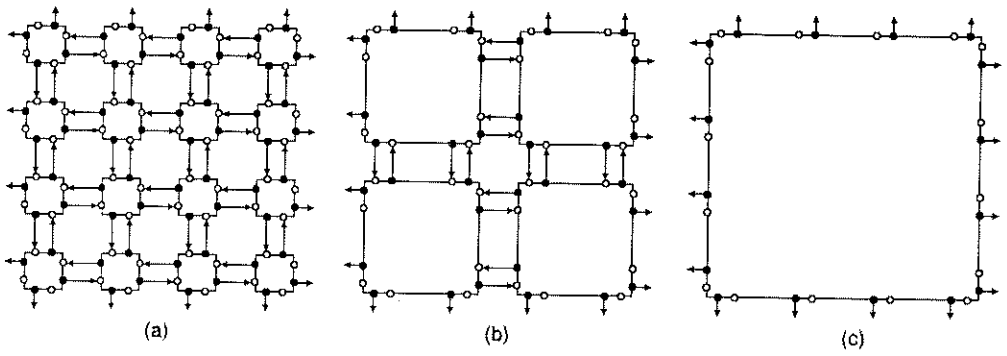


Figura 18: El procedimiento *Going_Out* propaga los datos de los ductos (mostrados como flechas) hacia los puertos externos; las figuras desde (a) hasta (c) nos muestran esta propagación en una secuencia de diferentes niveles de bloques.

las conexiones del ducto determinadas en la primera fase (a través del *cierre transitivo* de cada bloque de procesadores), para calcular así los datos que entrarán a cada bloque.

Este procedimiento realiza una iteración desde los bloques mayores hasta los menores, iniciando a partir del mosaico de $N \times N$ (toda la malla) y dividiéndolo en cuatro sub-mosaicos de $\frac{N}{2} \times \frac{N}{2}$, y así sucesivamente hasta llegar a los procesadores individuales, como puede apreciarse en la figura 19.

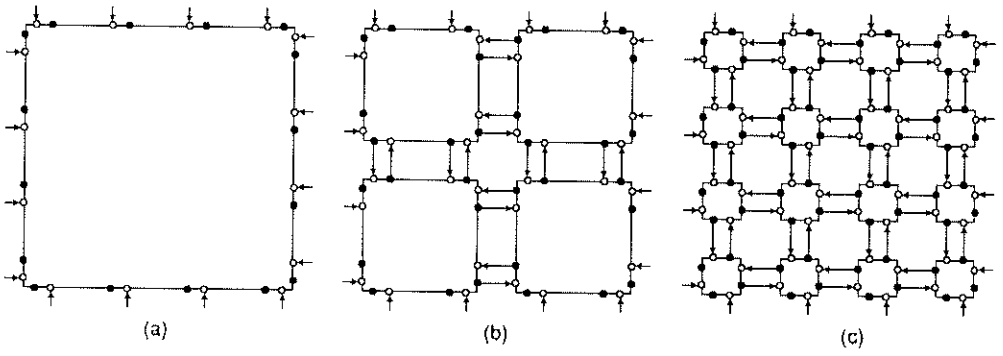


Figura 19: El procedimiento *Going_In* propaga los datos de los ductos (mostrados como flechas) hacia los puertos internos; las figuras, desde (a) hasta (c), nos muestran esta propagación en una secuencia de diferentes niveles de bloques.

En esta simulación se requiere un tiempo de $O(\log N)$, para ejecutar las multiplicaciones de las matrices booleanas, durante el cálculo del *cierre transitivo*; como se hizo

en la primera simulación [Trahan *et al.*, 1997] y se describe brevemente en la sección IV.1. Además, se requiere de un factor de tiempo $O(\log N)$ para las iteraciones; por ello, el tiempo total del algoritmo es $O(\log^2 N)$.

Capítulo V

Simulación propuesta

En el capítulo IV se hizo un compendio de las simulaciones existentes entre el DR-Mesh y el LR-Mesh. En el caso de la primera simulación diseñada [Ben-Asher *et al.*, 1991, Trahan *et al.*, 1998], por ser ésta indirecta y utilizar en el ínter modelos no reconfigurables (ver figura 12), el número de procesadores requeridos se incrementa bastante al ir pasando de un modelo a otro, por lo que finalmente se requiere de $O(N^{12})$ procesadores en el LR-Mesh. Sin embargo, la utilización del algoritmo *cierre transitivo* para la simulación la hace eficiente en tiempo, esto es, $O(\log N)$. Una máquina de este tipo no es viable de ser fabricada. En la segunda simulación, también indirecta, se simula al DR-Mesh sobre un R-Mesh, y después se simula al R-Mesh sobre el LR-Mesh.

La cantidad de procesadores requeridos es también elevada ya que en la primera parte se utiliza el *cierre transitivo*, aunque con menos recursos comparados con la primera simulación. Esto es en gran parte por ser una simulación indirecta de menos etapas. La tercera simulación, que también usa el *cierre transitivo*, es directa y los recursos requeridos se reducen bastante. Esto se debe a que ambos modelos tienen muchas características comunes que facilitan la simulación. Las diferencias son únicamente que el LR-Mesh tiene ductos no dirigidos y 4 puertos permitiendo 10 configuraciones distintas en ellos, mientras que el DR-Mesh tiene ductos dirigidos y 8 puertos, permitiendo 4140 configuraciones distintas. Con estos resultados se puede intuir que utilizar el algoritmo

del *cierre transitivo* no es ya la mejor opción. De aquí que la simulación propuesta en este trabajo no se basa en el *cierre transitivo.* En la simulación que se propone, la cantidad de recursos del modelo simulador es del mismo orden que en el modelo a simular (esto es $O(N^2)$ procesadores). El primer paso para diseñar esta simulación es asumir que el DR-Mesh es acíclico; esta suposición se hace como una primera aproximación ya que trabajar con ciclos dificulta enormemente la simulación. La simulación se ha dividido en dos partes para facilitar su descripción. La primera parte es una simulación aleatoria de un DR-Mesh sobre un R-Mesh (ver Figura 20). La segunda parte transforma al R-Mesh (sin ciclos) sobre el LR-Mesh (ver Figura 20).

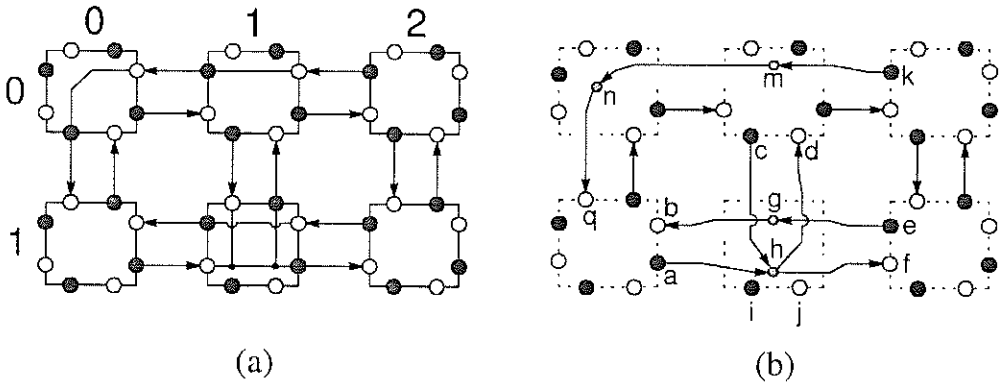
Figura 20: Etapas de la simulación propuesta.

Dos problemas básicos que hubo que superar para la realización de esta simulación fueron los siguientes:

- Diseñar una forma de simular los 8 puertos de un procesador del DR-Mesh así como sus conexiones internas en un LR-Mesh. Esto está directamente relacionado con el ancho de bisección en el procesador (la máxima cantidad de mensajes que se pueden transmitir simultáneamente sin que dos o más mensajes utilicen la misma ruta). Para emular un procesador del DR-Mesh en un bloque de procesadores LR-Mesh, se requiere tener el mismo ancho de bisección, y se necesitan al menos 4×4 procesadores del LR-Mesh.

- Diseñar un mecanismo para simular la direccionalidad del ducto del DR-Mesh en el LR-Mesh. Una forma es diseñar un algoritmo para propagar los datos segmento por segmento teniendo cuidado de propagar la información únicamente por los ductos permitidos en el DR-Mesh. Una técnica útil para lograr esto es el empleo de un algoritmo de contracción de árboles [JáJá, 1992].

Las conexiones del DR-Mesh se pueden modelar como un grafo dirigido. Cada nodo corresponde a un bloque de la partición del conjunto de puertos, y cada arista a la conexión entre dos bloques de puertos. Los grados de entrada/salida de cada nodo están limitados a cuatro. Como se explica más adelante, el problema de esta simulación se reduce a encontrar, para cada nodo i, cuáles son los nodos a los que se puede llegar desde i. Desde el punto de vista del DR-Mesh, equivale a obtener hasta que puertos se propaga un dato escrito por el bloque de puertos i. La figura 21b muestra el grafo dirigido equivalente del DR-Mesh de 2×3 de la figura 21a. El nodo h representa al conjunto de puertos $\{N_i, N_o, W_i, E_o\}$ del procesador (1,1). Similarmente, el nodo g representa el conjunto de puertos $\{W_o, E_i\}$ del mismo procesador. Los nodos i y j están aislados, ya que los puertos S_i y S_o no están conectados a ningún otro puerto. A continuación se explican brevemente las etapas de la simulación.



(a) (b)

Figura 21: Representación del DR-Mesh como un grafo. a) DR-Mesh de 2×3; b) Grafo equivalente del DR-Mesh.

Como el grafo equivalente del DR-Mesh no tiene ciclos, éste se puede modelar como un grafo acíclico dirigido (DAG). Un DAG se puede representar como una secuencia de nodos en donde las aristas dirigidas siempre van de izquierda a derecha (ver figura 22). Como el número de procesadores en el DR-Mesh es N^2, el número de nodos en la DAG es $O(N^2)$. En esta tesis se considera que la configuración más problemática en el DR-Mesh es cuando existen longitudes de ductos largos, esto es, ductos que cruzan $O(N^2)$ procesadores. Esto obedece a la necesidad de rastrear la propagación de un dato a través de este conjunto tan grande de procesadores como se muestra en el ejemplo de la figura 23.

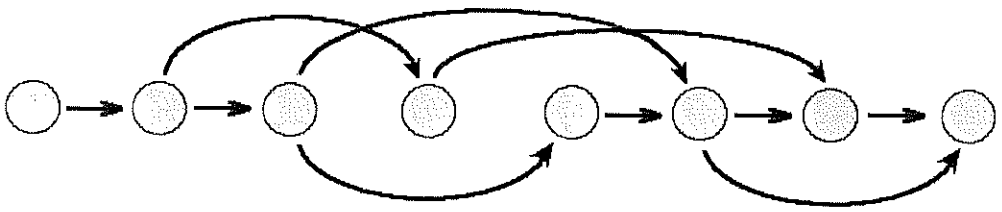


Figura 22: Grafo acíclico dirigido.

En este ejemplo, el procesador A escribe un dato en su puerto E_o, siendo el procesador B el más lejano de A. Los ductos están configurados de tal forma que el dato tiene que pasar por $O(N^2)$ procesadores. En el caso del DR-Mesh la transferencia del dato se logra en un tiempo constante.

Una forma de simular a un DAG (y a su vez a un DR-Mesh) en un R-Mesh, es trasladando a una malla de 4×4 procesadores R-Mesh, al conjunto de nodos de cada procesador DR-Mesh. En esta malla de 4×4 procesadores, se pueden simular las cuatro aristas de entrada y las cuatro de salida (ver figura 24a). Para las aristas de entrada se usan ductos horizontales, y para las aristas de salida ductos verticales; así se puede configurar cualquier nodo (en el peor de los casos, cada nodo representa un procesador completo del DR-Mesh).

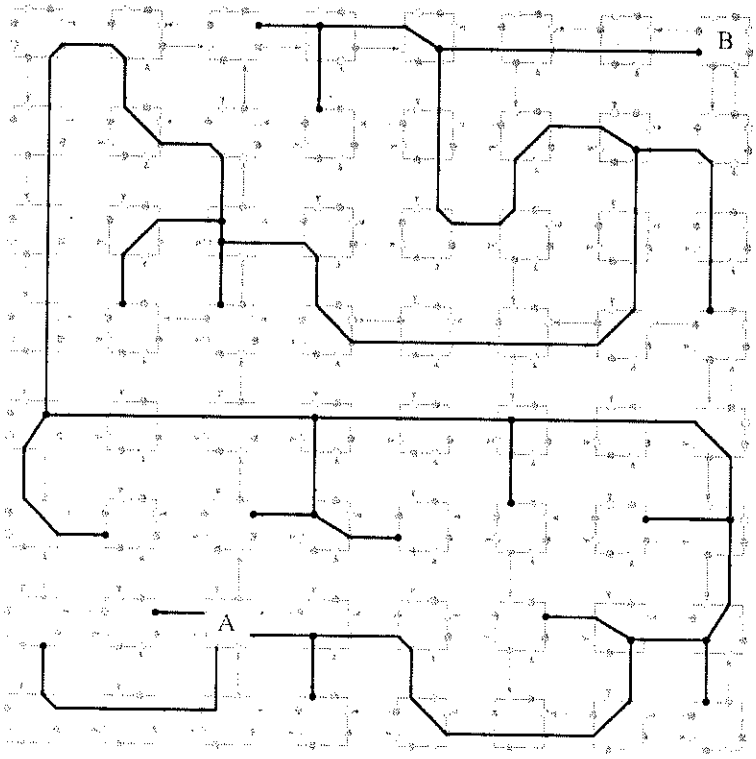


Figura 23: Posible trayectoria en la propagación de un dato en un DR-Mesh.

Por ejemplo, para conectar el puerto de entrada E_i con los puertos de salida E_o, S_o y W_o, sólo se tiene que conectar los puntos de cruce en los procesadores (1,3), (1,1) y (1,0), respectivamente (ver figura 24b). En un procesador del DR-Mesh se pueden conectar cualquiera de sus puertos entre sí. La malla de 4×4 procesadores R-Mesh no permite conectar exclusivamente a ductos horizontales entre sí, o exclusivamente a ductos verticales entre sí; esto no es problema para la simulación ya que en la práctica estas conexiones no tienen beneficio alguno.

El problema ahora es simular la direccionalidad de los ductos del grafo. Para solucionar este problema, se conecta en cada malla de 4×4 solamente una arista de entrada a la vez, con la(s) respectiva(s) arista(s) de salida, según sea el caso. De esta forma, se evita que un dato entrante se propague a través de otra arista de entrada, dado que los ductos en los procesadores del R-Mesh son bidireccionales y lo que se

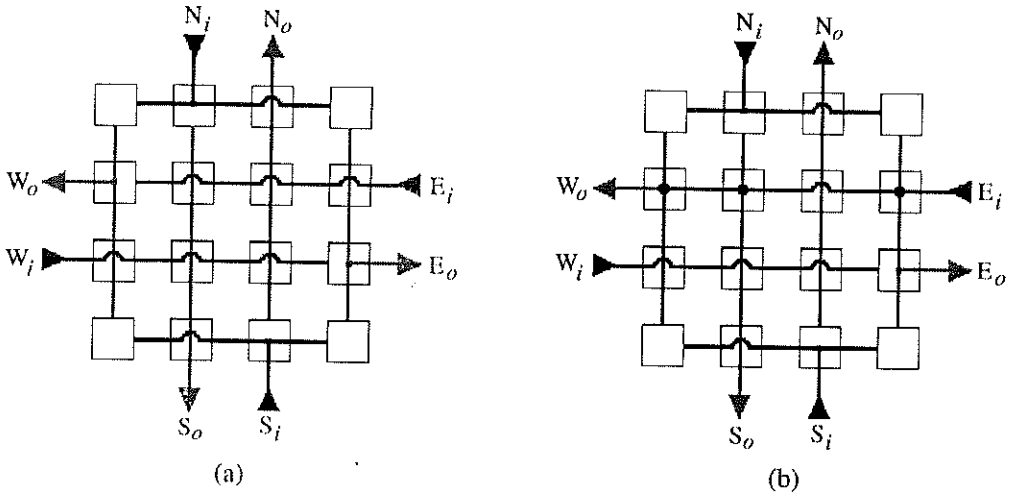


Figura 24: Malla 4×4 R-Mesh para simular a un procesador del DR-Mesh.

pretende es simular ductos dirigidos. La figura 25 ilustra como las aristas de salida se conectan juntas y se hacen conexiones aleatorias entre éstas y sólo una arista de entrada.

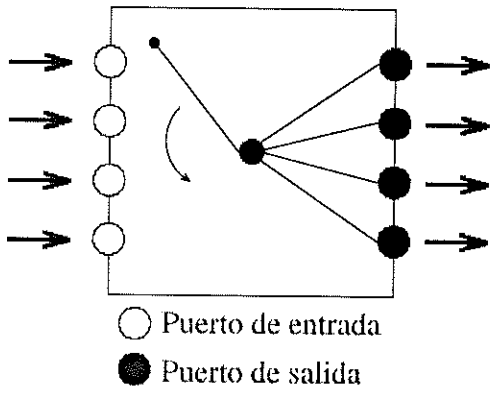


Figura 25: Simulación de direccionalidad en la malla 4×4 R-Mesh.

La simulación de un DR-Mesh sobre un LR-Mesh tiene una gran importancia práctica, ya que en teoría es mucho más sencillo construir un LR-Mesh que un DR-Mesh; además, si se permite incrementar en un factor de 4 al LR-Mesh, se pueden reemplazar las escrituras concurrentes por escrituras exclusivas sin penalización en tiempo [Fernández-Zepeda *et al.*, 1999]; ésto hace al modelo simulador aún más viable. En la

sección siguiente, se describe la simulación del DR-Mesh sobre el R-Mesh. Posteriormente, en la sección V.3, se describe la simulación del R-Mesh sobre el LR-Mesh.

V.1 Algoritmo de simulación del DR-Mesh en R-Mesh

Para simular a un procesador del DR-Mesh, se debe simular cada una de sus cuatro etapas de cada ciclo de máquina.

- La primera etapa es configurar un patrón de conexiones en los puertos, que se logra con la malla de 4×4 procesadores R-Mesh (ver figura 24), y se establece en el Lema 1 (sección V.2).

- La segunda etapa es cuando el procesador DR-Mesh escribe en sus puertos de salida. Considerando que los ductos en los procesadores DR-Mesh son dirigidos, en la malla de 4×4 procesadores R-Mesh se simula la direccionalidad conectando aleatoriamente un sólo puerto de entrada a la vez con todos los de salida; como se aprecia en la figura 25. De esta forma se evita que un dato se propague de regreso por otro de puerto de entrada.

- La tercera etapa consiste en leer de los ductos conectados a los puertos de entrada, lo cual en la simulación es trivial una vez superada la etapa anterior.

- La cuarta etapa consiste en realizar las operaciones aritméticas o lógicas usando el DR-Mesh sus datos locales. Una vez que el nodo tiene el dato, cualquier procesador del R-Mesh de la malla de 4×4 correspondiente puede ejecutar dichas operaciones.

El algoritmo que realiza la simulación del DR-Mesh sobre el R-Mesh es un proceso iterativo que consiste de cuatro etapas que se ejecutan en forma secuencial en cada procesador (ver el pseudocódigo en la figura 26). Los datos de entrada al algoritmo son

Input:
- Patrón de conexiones del DR-Mesh (en cada procesador).
- El dato a transmitir en cada nodo de la etapa inicial.

Output:
- Un árbol con únicamente los nodos que reciben el dato.
- Dato en cada puerto de entrada al finalizar el ciclo de escritura.

Configura_puertos
For $i \leftarrow 1$ to β realiza en paralelo
Configura_dirección
Propaga_valores
Poda_nodos
End

Figura 26: Pseudocódigo del algoritmo de simulación del DR-Mesh en el LR-Mesh.

todas las conexiones existentes en los procesadores y el dato a transmitir (si existe) en cada puerto. A continuación, se describen cada una de las etapas de esta simulación.

1. La rutina *Configura_puertos* fija las conexiones internas de cada conjunto de 4×4 procesadores en el R-Mesh (ver figura 24), de acuerdo a la configuración correspondiente de cada nodo del grafo equivalente del DR-Mesh, y las interconexiones entre los conjuntos de 4×4 procesadores. De esta manera, se consigue empotrar el DAG del DR-Mesh en el R-Mesh.

2. La rutina *Configura_dirección* desconecta en cada bloque de puertos todos los puertos de entrada, excepto uno (escogido al azar) para evitar que un dato que llegue por un puerto de entrada se propague por otro puerto de entrada (ver figura

25). Esto, con la finalidad de que el grafo sea dirigido y simular la direccionalidad de los ductos del DR-Mesh.

3. La rutina *Propaga_valores* simula la propagación de los datos escritos por los puertos en los ductos. Como cada nodo sólo conecta una arista de entrada con todas las salidas, en caso de que llegue un dato por la arista de entrada escogida, éste se propaga sin conflicto por todas las aristas de salida a nodos subsecuentes. En caso de que un dato llegue por una arista de entrada que no esté conectada con las salidas, el nodo tomará el dato y lo transmitirá por sus aristas de salida en el siguiente ciclo.

4. La rutina *Poda_nodos* remueve a los nodos que no reciben ni transmiten dato alguno en todo el proceso. Como no se conoce de antemano la identidad de estos nodos, este proceso se efectúa en forma iterativa, removiendo grupos de nodos a la vez. En general, los nodos que se remueven son todos aquellos con grado de entrada cero y que no transmiten dato, y todas las cadenas lineales de nodos que estén conectadas a éstos y que no transmitan datos, así como a todas las aristas o enlaces que de ellos emanen.

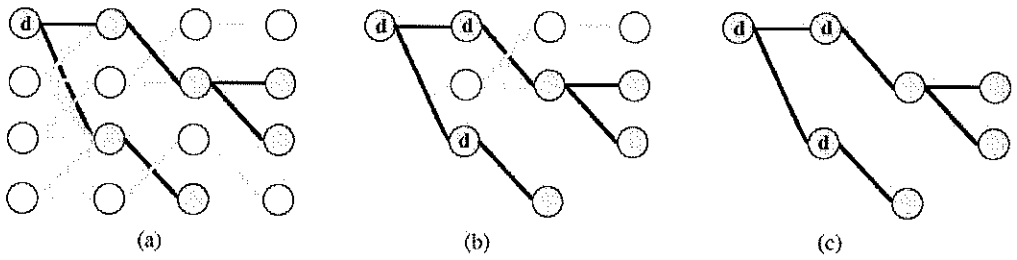


Figura 27: Ejemplo del podado de nodos.

En la figura 27, se muestra un pequeño ejemplo de la rutina de podado; en la primera etapa (figura 27a) se tiene un conjunto de nodos y un dato d a transmitirse de izquierda a derecha. Las aristas en negro son las trayectorias por donde el dato d debe

propagarse al finalizar la simulación. Las aristas en color claro son aquellas que no propagan ningún dato y por lo tanto se eliminan. Después del primer ciclo de podado (figura 27b), se eliminan seis nodos y sus respectivas aristas. Tres de estos nodos se eliminan por tener grado de entrada cero y no tener ningún dato a transmitir; los otros tres nodos se eliminan por pertenecer a una cadena lineal que proviene de un nodo con grado de entrada cero y que no contiene dato. En el siguiente ciclo de podado para este ejemplo, se eliminan tres nodos, uno por tener grado de entrada cero y ningún dato a transmitir, y los otros dos, por pertenecer a una cadena lineal proveniente del nodo anteriormente descrito, asegurándose, de esta manera, la transmisión del dato a la siguiente columna de nodos. Los nodos que se eliminan después del segundo ciclo de podado son los de color blanco en la figura 27b, y sus respectivas aristas que están marcadas en color claro, quedando finalmente el árbol que se muestra en la figura 27c.

Este algoritmo corre en el R-Mesh. A pesar de que se requieren 16 procesadores en el R-Mesh para simular un nodo del grafo equivalente del DR-Mesh, ambos modelos tienen el mismo orden de procesadores. A continuación, se comprueba el funcionamiento de este algoritmo.

V.2 Comprobación del algoritmo

En esta sección, se comprueba el algoritmo a través de dos lemas. El primer lema está relacionado con la simulación del patrón de conexiones del DR-Mesh. El segundo lema está relacionado con la simulación del ciclo de escritura. Como se mencionó anteriormente, el ciclo de lectura y ejecución son triviales, una vez simulados los primeros dos.

Lema 1 *Un conjunto de* 4×4 *procesadores R-Mesh puede simular el conjunto de conexiones internas de un procesador DR-Mesh.*

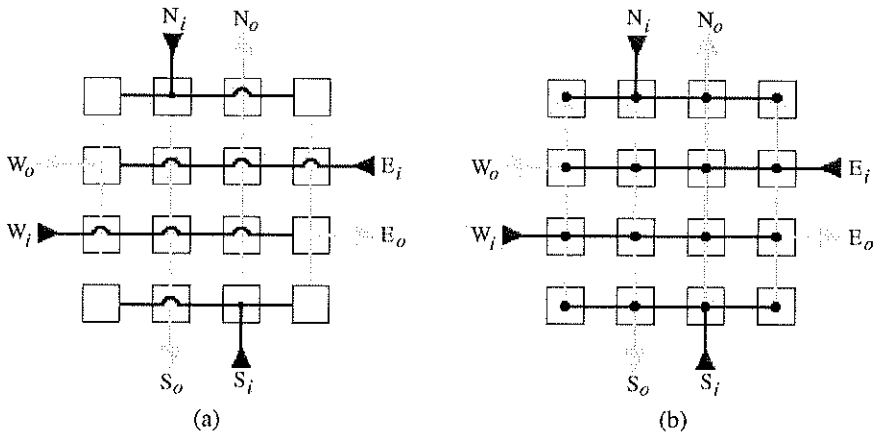


Figura 28: Malla 4×4 R-Mesh para simular a un procesador del DR-Mesh. a) malla sin conexión; b) malla en donde todos los puertos están conectados entre sí.

DEMOSTRACIÓN. Como se observa en la figura 28a, los ductos horizontales de la malla de 4×4 están conectados a los puertos de entrada y los ductos verticales a los puertos de salida. La interconexión apropiada de ductos horizontales con ductos verticales permite comunicar a cualquier puerto de entrada con cualquier puerto de salida. De hecho, esta configuración de ductos es equivalente a un *crossbar* (interconexión en la que cada entrada está conectada a cada salida a través de una trayectoria que contiene un solo nodo de conmutación) [Duato *et al.*, 1997]. En la figura 28b, se tienen todos los ductos horizontales (color oscuro) interconectados con todos los ductos verticales (color claro) para comunicar a todos los puertos de entrada con todos los puertos de salida; cualquier otro caso requiere de menos puntos de interconexión. Los únicos casos que no es posible simular es cuando el conjunto de puertos conectados es exclusivamente de entrada o exclusivamente de salida; de cualquier forma, estos dos tipos de conexiones son inútiles en la práctica como se explica a continuación.

Supóngase que en la figura 29a, el procesador DR-Mesh transmite datos por sus puertos N_o y W_o, dado que los ductos son dirigidos; los datos se propagan del puerto hacia afuera del procesador y no hacia adentro. Si internamente ambos puertos estu-

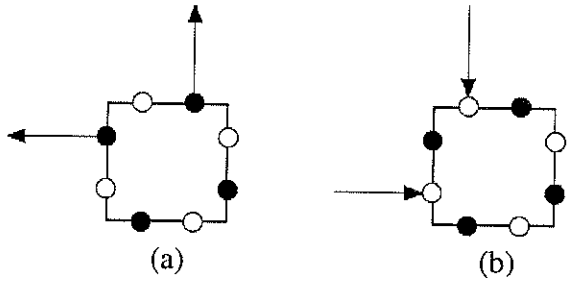


Figura 29: Efecto de conexiones entre puertos; a) la conexión exclusiva de puertos de salida equivale a dejarlos desconectados; b) la conexión exclusiva de puertos de entrada equivale a dejarlos desconectados.

viesen conectados, el efecto sería el mismo. De manera similar, supóngase que en la figura 29b, el procesador DR-Mesh recibe dos datos simultáneamente por sus puertos de entrada N_i y W_i. Tan pronto llegue el dato al puerto, éste lo lee, y lo que suceda con el dato posteriormente (en caso de que el ducto continúe) ya no tiene impacto en ese puerto, ya que el ducto es dirigido. Así que la conexión exclusiva de varios puertos de entrada o de salida es equivalente a dejarlos desconectados. □

Note que la simulación de varios procesadores DR-Mesh se puede hacer con un número igual de mallas de 4×4 procesadores. Estas mallas se pueden conectar como se muestra en la figura 30.

Lema 2 *Si durante la simulación de un DR-Mesh de $N \times N$, existe una trayectoria del puerto a al puerto b, cualquier dato escrito en el puerto a llegará al puerto b a lo más en $O(N^2)$ u.t.*

DEMOSTRACIÓN. Supóngase que el nodo x_a (del grafo equivalente del DR-Mesh) contiene al puerto a y el nodo x_b contiene al puerto b. Supóngase que existe una ruta dirigida del nodo x_a hacia el nodo x_b, y esta ruta pasa por los nodos $x_1, x_2, \ldots, x_{k-1}, x_k$ (como se observa en la figura 31). Supóngase además que el nodo x_a escribe un dato en su arista de salida; existen dos posibles casos:

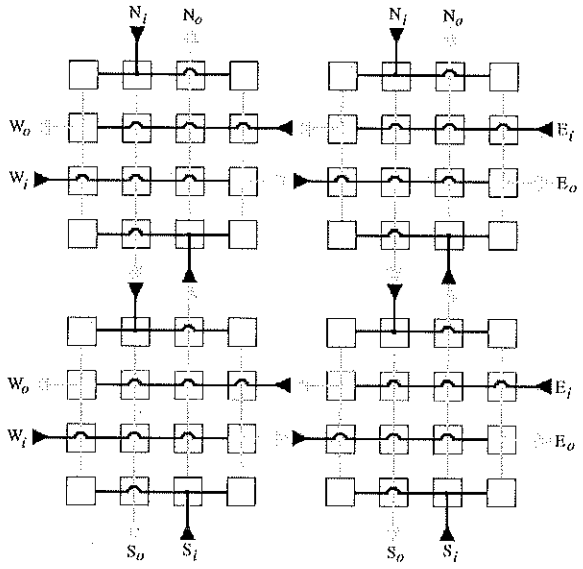


Figura 30: Representación de un procesador DR-Mesh por una malla de 4×4 procesadores R-Mesh.

caso a) El nodo x_1 conecta la entrada proveniente del nodo x_a a todas sus salidas, propagando así el dato de entrada al nodo x_2.

caso b) El nodo x_1 conecta una entrada distinta a la proveniente de x_a con todas sus salidas. Para este caso, el dato proveniente de x_a no se propaga al nodo x_2. El algoritmo permite a los nodos leer todos sus puertos de entrada en cada iteración, así que el nodo x_1 lee el dato proveniente del nodo x_a. En la siguiente iteración, el nodo x_1 propaga dicho dato por todas sus salidas y llega así al nodo x_2.

Como puede verse en los dos casos anteriores, el dato proveniente del nodo x_a logra propagarse al menos al nodo x_1, con un retardo máximo de un paso.

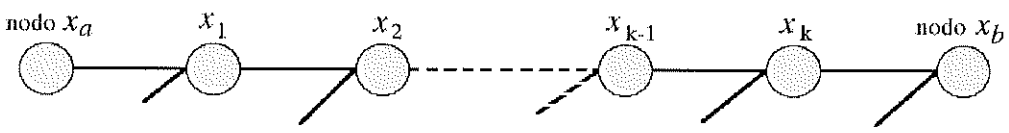


Figura 31: Trayectoria entre el nodo x_a y el nodo x_b.

Este proceso se repite para los nodos $x_1, x_2, \ldots, x_{k-1}, x_k$ quedando asegurada así la llegada del dato al nodo x_b a lo más en $k+1$ pasos. Si la trayectoria entre los nodos x_a y x_b pasa por la mayoría de los procesadores DR-Mesh (como es en el caso de la figura 23), entonces el valor de k que representa la cantidad de nodos es $O(N^2)$, siendo éste el tiempo máximo requerido para terminar la simulación. □

Los dos lemas anteriores apoyan que la simulación de la configuración del DR-Mesh y del ciclo de escritura son correctos. Una vez generado el dato final en cada arista, cada nodo lee este dato finalizando el ciclo de lectura. El ciclo de ejecución lo puede realizar cualquier procesador de cada nodo. De esta forma queda comprobada la simulación de un ciclo de máquina arbitrario de un DR-Mesh acíclico sobre un R-Mesh utilizando el mismo orden de procesadores ($O(N^2)$).

A continuación se describe la simulación del R-Mesh sobre el LR-Mesh.

V.3 R-Mesh sobre LR-Mesh

La mejor simulación de un R-Mesh de $N \times N$ sobre un LR-Mesh de $N \times N$ corre en $O(logN)$ [Fernández-Zepeda, 1999]. Aunque en el presente trabajo se utiliza un R-Mesh, el grafo equivalente del patrón de conexiones utilizado en la presente simulación es un bosque; este hecho se muestra a continuación. Sea n_j un nodo arbitrario de la etapa i; sea n_k el nodo de la etapa $i-1$ que esta conectado al nodo n_j (recuerde que el nodo n_j sólo puede estar conectado a lo más a un nodo de la etapa $i-1$). Sean $\{s_{j,1}, s_{j,2}, s_{j,3}, s_{j,4}\}$ las salidas del nodo n_j. Estas salidas, a su vez, pueden ir conectadas a los nodos n_1, n_2, n_3 y n_4 de la etapa $i+1$ (ver figura 32).

El nodo n_k representa al padre de n_j y el nodo n_j representa a su vez al padre de los nodos n_1, n_2, n_3 y n_4. Es importante notar que cada nodo tiene a lo más un padre,

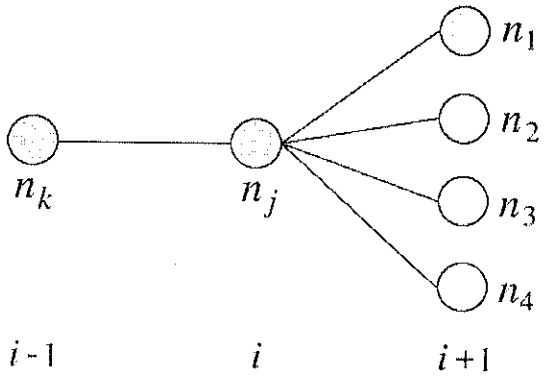


Figura 32: Relación entre los nodos de un árbol.

con excepción de los nodos en la primera etapa, o algún otro que no tenga conexiones en la etapa anterior. Por lo tanto, esta conexión es equivalente a un bosque en el caso general. Como el patrón de conexiones del R-Mesh utilizado no tiene ciclos, se puede simular en un LR-Mesh, con el mismo orden de procesadores y en un tiempo $O(1)$ [Ben-Asher *et al.*,1992]. Este proceso se describe brevemente a continuación.

La configuración del R-Mesh se puede simular reemplazando cada arista del árbol por dos aristas (ver figura 33), y construir después un Tour de Euler para linealizar los ductos [Fernández-Zepeda *et al.*, 1999].

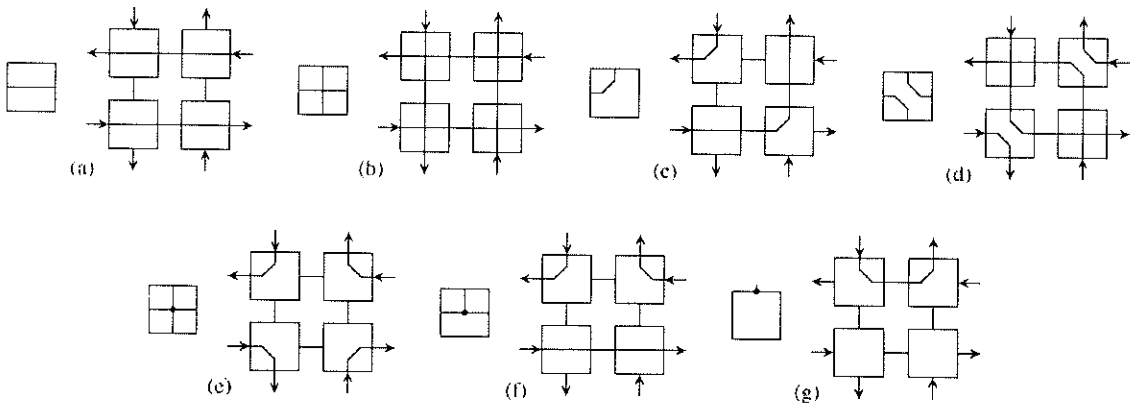


Figura 33: Grupo equivalente de configuraciones LR-Mesh por cada procesador R-Mesh; a-d) lineal; e,f) no-lineal; g) punto terminal.

Sea $T = (V, E)$ un árbol y $T' = (V, E')$ el grafo dirigido obtenido de T, donde cada arista $(u, v) \in E$ se reemplaza por dos arcos $\langle u, v \rangle$ y $\langle v, u \rangle$. Como el grado de entrada de cada nodo de T' es igual a su grado de salida, T' es un grafo Euleriano; es decir, existe una trayectoria cerrada que atraviesa a cada arista del grafo exactamente una vez. De esta forma, el circuito Euler de T' puede utilizarse para optimizar el computo paralelo de muchas funciones de T [JáJá, 1992]. Con lo anteriormente expuesto, se demuestra el siguiente teorema.

Teorema 3 *Cualquier ciclo de máquina de un DR-Mesh acíclico de $N \times N$ procesadores, se simula sobre un LR-Mesh de $O(N) \times O(N)$ procesadores en $O(N^2)$ unidades de tiempo.*

El algoritmo asegura que el tiempo de ejecución es, en el peor de los casos, $O(N^2)$ u.t. Sin embargo, por todas las pruebas llevadas a cabo, se puede determinar que, para el caso promedio, el tiempo de ejecución de la simulación es mucho más corto.

En la sección V.3 se analiza la simulación del R-Mesh en el LR-Mesh, la cual se logra en un tiempo constante por la forma como se implanta el algoritmo, utilizando el mismo orden de procesadores. Por lo tanto, el tiempo total de la simulación está acotado por el tiempo de ejecución de la primera etapa (DR-Mesh en R-Mesh), descrita en la sección V.1, cuyo tiempo promedio se evalúa en el capítulo VI.

Capítulo VI

Estimación del tiempo de ejecución promedio

En el capítulo V se analiza al algoritmo propuesto y se demuestra su validez. Aunque el tiempo de ejecución de éste es $O(N^2)$ en ciertas condiciones poco probables de presentarse; para el caso promedio el tiempo de ejecución es posiblemente menor a N como lo muestran los resultados numéricos. La demostración analítica es más compleja y se toma como línea de investigación futura.

Como se menciona en el capítulo V, se considera que para esta simulación, la configuración más problemática en el DR-Mesh es cuando los ductos son largos, esto es cuando los ductos cruzan $O(N^2)$ procesadores. Lo anterior se debe a que el dato a transmitirse se tiene que propagar a lo largo de todo el ducto (ver figura 23); en el DR-Mesh, esta propagación se logra en un tiempo constante, mientras que en el R-Mesh la propagación es más lenta por la segmentación de ductos que hace el algoritmo. Si se determina el tiempo de propagación promedio del dato para el caso crítico en un R-Mesh, se deduce que el tiempo de propagación promedio para cualquier otro caso es menor, o a lo más igual a ese tiempo promedio requerido para el caso crítico. Para simular dicho caso, se construye una malla como se muestra en la figura 34. Esta malla es una representación idealizada del grafo equivalente del DR-Mesh para el caso cuando las longitudes de los ductos son largas, esto es $O(N^2)$. El número de etapas en esta

malla es $O(N^2)$ y el ancho es constante. Cada círculo de esta malla representa un nodo del grafo equivalente del DR-Mesh.

Por simplicidad, este modelo se construye de forma regular, y sólo se permiten conexiones de izquierda a derecha y entre nodos de etapas subsecuentes. Las conexiones entre etapas se generan aleatoriamente; el grado de entrada y de salida puede variar de 0 a 4. Variando el grado del nodo se pueden tener configuraciones densas o dispersas, así se puede simular la malla para diversos escenarios.

El algoritmo que simula el comportamiento de esta malla es equivalente al algoritmo de simulación que corre en el R-Mesh. Se supone que un nodo en la primera etapa de la malla escribe un dato en sus aristas de salida. El algoritmo reproduce la forma en que este dato se propaga a través de la malla, contabilizando el tiempo de propagación.

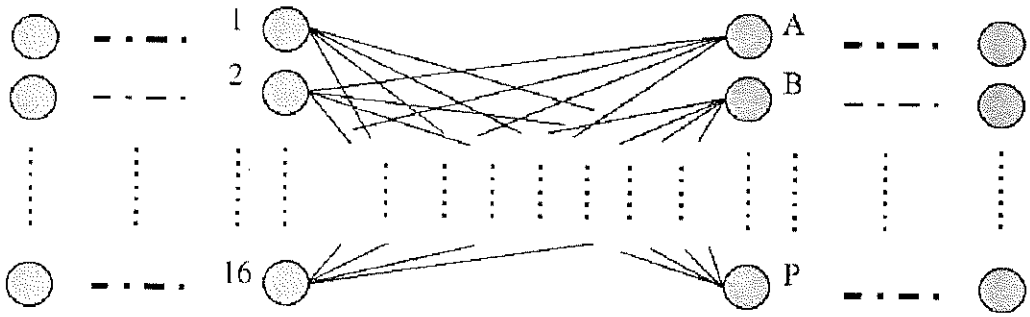


Figura 34: Modelo simplificado del grafo equivalente del DR-Mesh.

En los experimentos preliminares se usa $N^2 = 10000$, el ancho de la malla se toma como 16 y el programa se corre con 1000 configuraciones distintas. En cada una de las 1000 corridas, se fija el grado de salida de cada nodo a un valor específico. En la gráfica 35, se muestran los resultados a manera de ejemplo de una prueba; el eje y muestra el número de experimentos en el que todos los procesadores que deben recibir dato lo recibieron, mientras que el eje x indica el número de intentos requeridos para

terminar de propagar el dato. El grado de salida promedio en cada nodo es 1.33. Como puede verse, se necesitaron a lo más 11 pasos (aunque en promedio fue menor) para completar el proceso.

Es conveniente aclarar que es posible que muchos procesadores, que no van a recibir dato, aún estén esperando una señal de confirmación de que no lo van a recibir; por lo tanto, la simulación no necesariamente ha terminado.

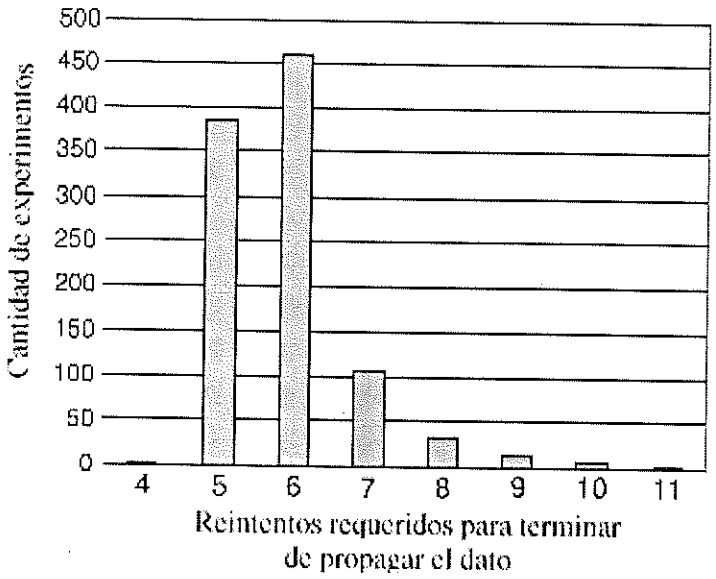


Figura 35: Gráfica que muestra la propagación del dato con 1.33 enlaces en promedio por nodo.

La gráfica 36 muestra, para el mismo ejemplo, el número de pasos en los que la simulación terminó. El eje y muestra el número de experimentos en el que todos los procesadores terminaron su trabajo, mientras que el eje x indica el número de reintentos requeridos para terminar la simulación. Como puede apreciarse, se requirieron a lo más 13 pasos, lo cual es mucho menor que N, que en este caso particular, es igual a 100.

En ciertas condiciones particulares, pueden requerirse hasta $O(N^2)$ pasos para terminar la simulación. Este caso tan particular, se debe al efecto de aquellos nodos que no reciben dato. Si cada nodo siempre conecta la entrada que no contiene dato con

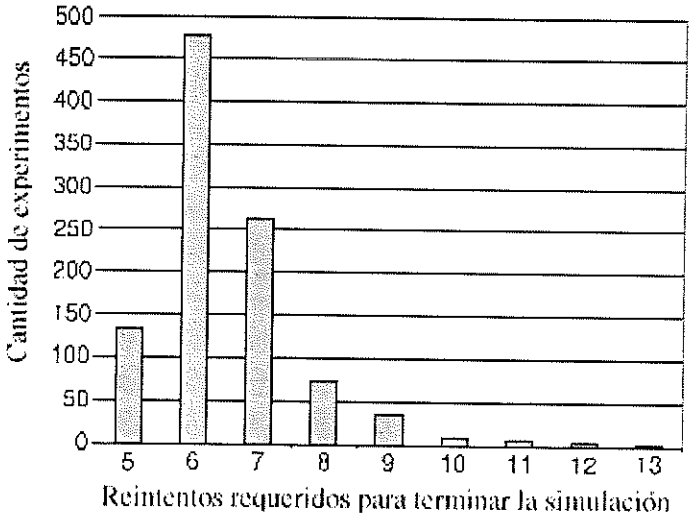


Figura 36: Gráfica que muestra la terminación del trabajo de todos los procesadores con 1.33 enlaces en promedio por nodo.

las salidas, la propagación del dato es lenta ya que el dato se propaga un nodo por ciclo. Si este DAG consta de $O(N^2)$ etapas, entonces se requiere de $O(N^2)$ pasos para asegurar la terminación de la simulación.

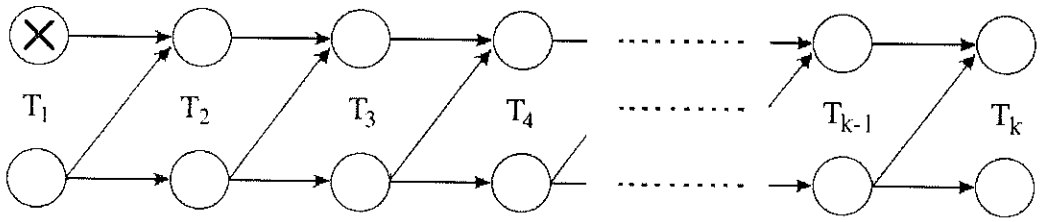


Figura 37: Ejemplo de propagación problemática.

Ejemplo: La figura 37 muestra un DAG con K etapas. El dato está inicialmente en el nodo superior de la primer etapa (el dato se representa como una X). La ruta de propagación del dato en este caso es hacia todos los nodos superiores de las etapas $T_2, T_3, \ldots, T_K$. Si el nodo superior en la etapa T_2 aleatoriamente se conecta a la arista de entrada proveniente del nodo inferior de la etapa T_1, entonces no se transmite el

dato a la etapa T_3, pero el nodo superior de la etapa T_2 lee el dato por estar éste presente en una de sus aristas de entrada, y lo retransmite en el siguiente ciclo. Si este problema se repite en cada etapa, entonces el algoritmo requiere de $K-1$ ciclos para terminar de propagar el dato. Nótese que para este caso, el algoritmo de podado no acelera la transmisión del dato. Nótese además que la probabilidad de que ésto ocurra es muy baja, $\left(\frac{1}{2}\right)^{K-1}$, para este ejemplo. Las figuras 38 y 39 representan los resultados para otro ejemplo con un promedio de enlaces mayor. Al comparar estas gráficas con las de la primera prueba, se puede apreciar una similitud en su comportamiento.

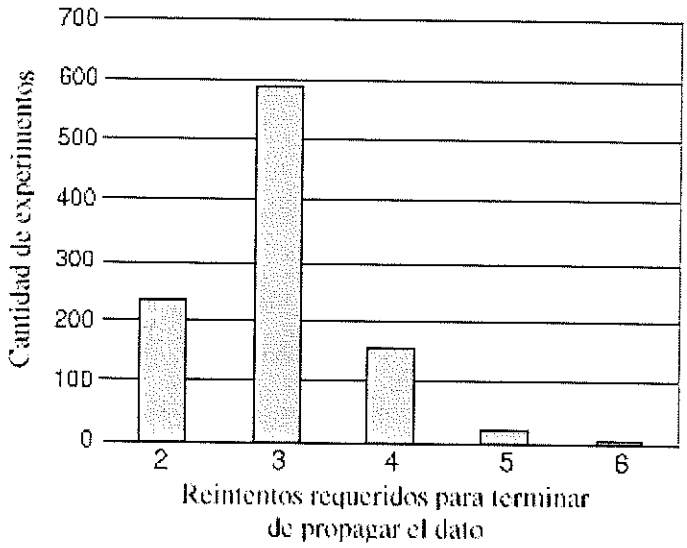


Figura 38: Gráfica que muestra la propagación del dato con 2.51 enlaces en promedio por nodo.

El problema principal que se tiene ahora es cómo hacer para que los procesadores que no van a recibir dato, se den cuenta de este hecho de una manera más rápida. El tiempo de terminación de este algoritmo está limitado por el tiempo de terminación del podado. Aunque el dato se propague en un sólo paso a todos los nodos correspondientes, la malla no puede saber que ya terminó el algoritmo mientras haya procesadores que no sepan si van a recibir dato, o si van a ser podados. De manera inversa, una vez que

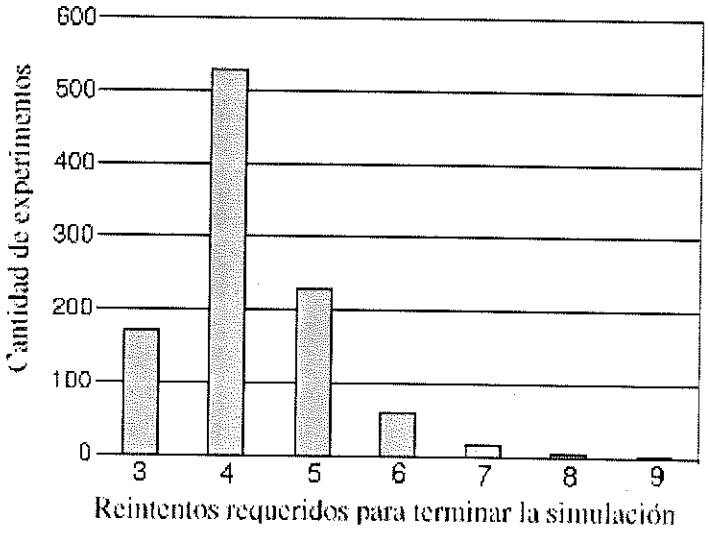


Figura 39: Gráfica que muestra la terminación del trabajo de todos los procesadores con 2.51 enlaces en promedio por nodo.

ya se podaron todos los nodos que nunca iban a recibir dato alguno, la propagación del dato a los nodos restantes se logra en un tiempo $O(1)$.

La figura 40 ilustra este caso; el dato a propagar se encuentra en la etapa inicial T_1; a los nodos de la etapa T_2 no les queda otra opción que conectar sus salidas a la única arista de entrada que tienen, la proveniente del nodo T_1. Los nodos de la etapa T_2 no sólo leen el dato, sino que al mismo tiempo lo dejan pasar a la siguiente etapa, la T_3.

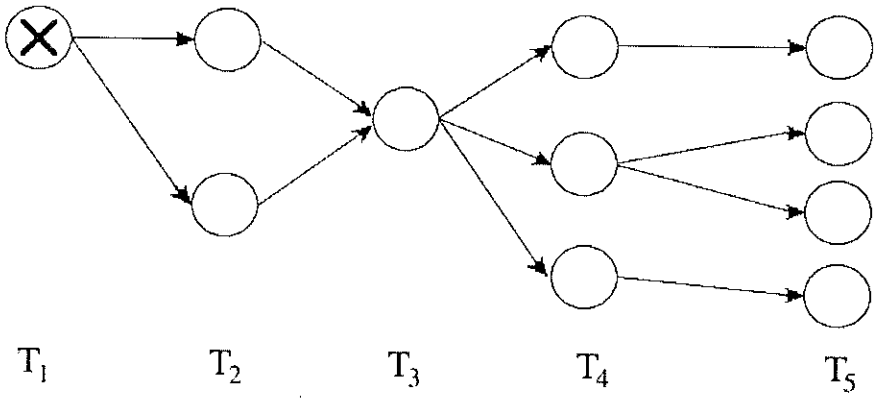


Figura 40: Nodos restantes después de completado el podado.

El único nodo de la etapa T_3 tiene dos aristas de entrada, no importa cual seleccione para conectarse; el dato lo deja pasar hacia sus tres aristas de salida a la etapa T_4. Este proceso continúa en el resto de la malla y toma un sólo paso.

En el algoritmo propuesto, antes de cada ciclo de podado hay un ciclo de propagación de dato; es posible que el dato pueda terminar de propagarse aún cuando no se haya terminado de podar a todos los nodos. Por lo cual, el tiempo de ejecución de este algoritmo está limitado a su tiempo de podado.

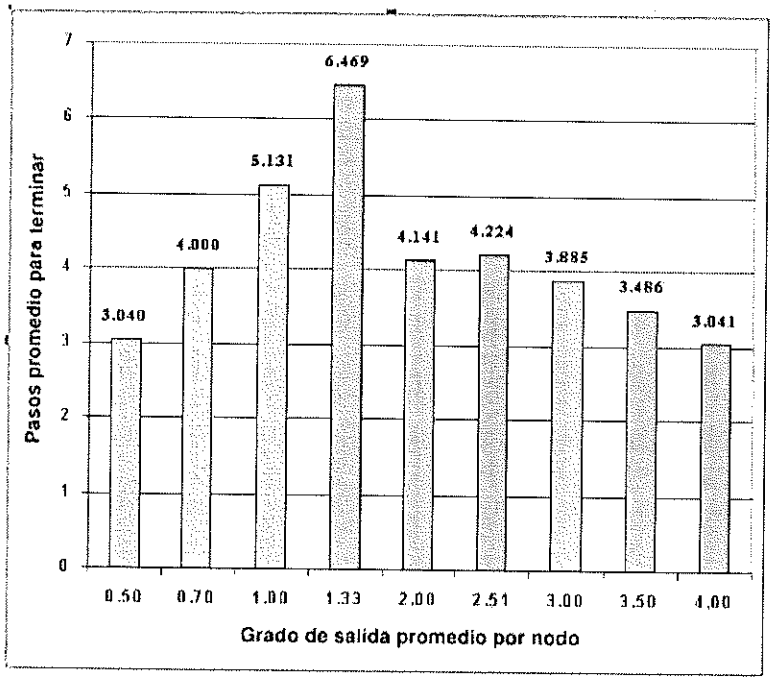


Figura 41: Gráfica que muestra los resultados para pruebas hechas con distintos promedios en enlaces.

En la figura 41 se muestran los resultados de pruebas hechas con diferentes promedios de enlaces. Cada prueba consiste de mil experimentos, con un mismo valor promedio de enlaces pero con diferentes patrones de conexiones. Como puede observarse, el caso que requirió más pasos en promedio (6.469 pasos) para terminar la

simulación es el que tiene 1.33 enlaces en promedio por nodo. Los casos más rápidos corresponden a promedios de 0.5 y 4 enlaces por nodo, que terminaron la simulación en 3.040 y 3.041 pasos en promedio, respectivamente.

Capítulo VII

Conclusiones y trabajo futuro

Es ya muy conocido que en el computo paralelo, los modelos con ductos reconfigurables son computacionalmente más poderosos que otros modelos paralelos no reconfigurables como la PRAM. Esto ha motivado el desarrollo de muchas investigaciones al respecto. La simulación del modelo DR-Mesh en el modelo LR-Mesh tiene una gran importancia práctica ya que permite trasladar todos los algoritmos que han sido diseñados para el modelo DR-Mesh al modelo LR-Mesh. Por ello, se han desarrollado ya algoritmos de simulación entre estos modelos. Todos los algoritmos anteriores al propuesto aquí hacen uso del *cierre transitivo*, la cual para su cálculo requiere de muchos recursos si se quiere realizar rápidamente. Si se emplean únicamente modelos con ductos reconfigurables para la simulación, se hace a ésta más eficiente en recursos, ya que se aprovechan al máximo las ventajas de la reconfiguración. Por lo anterior expuesto, se decidió para esta simulación no usar el *cierre transitivo*, y aunque es una simulación indirecta, se emplean sólo modelos con ductos reconfigurables.

VII.1 Conclusiones

El algoritmo propuesto simula un ciclo de máquina arbitrario de un DR-Mesh acíclico de $N \times N$ procesadores sobre un LR-Mesh de $O(N \times N)$ procesadores a lo

más en $O(N^2)$ u.t.

Para este algoritmo, como puede apreciarse, se requiere del mismo orden de procesadores en el modelo simulador que en el modelo simulado, lo cual es el logro más significativo, aunque el DR-Mesh considerado es un modelo restringido. La simulación anterior más eficiente en recursos requiere de $O\left(\frac{N^4}{log^2 N}\right)$ para un modelo sin restricciones. Aunque el tiempo de ejecución del algoritmo propuesto es $O(N^2)$, la cota superior es mucho menor para el caso promedio, como se explica en la sección VI. Mediante un programa de computadora, se corrió al algoritmo miles de veces con configuraciones distintas para $N^2 = 10000$, obteniendose aproximaciones numéricas para diferentes promedios de enlaces. Como se aprecia en la figura 41, se requirieron unos cuantos pasos (menos de 7, que es mucho menor que N) para terminar la simulación. También, en la misma sección VI, se muestra porqué es tan poco probable que el algoritmo tarde $O(N^2)$ pasos para terminar. Este caso sólo puede suceder cuando un dato se propaga a través de un ducto que cruza $O(N^2)$ procesadores. Se intuye que esta cota superior puede ser más chica en promedio ya que muchas veces los ductos tienen una longitud más corta.

El tiempo de terminación del algoritmo está acotado por el tiempo de terminación del podado, ya que el podado es independiente del avance del dato. El avance del dato se beneficia con los ciclos de podado, ya que las opciones de conexión se reducen (ver sección V.1). Una vez acabado el podado, el dato termina de propagarse a los nodos restantes en un paso, si es que no terminó antes. Los resultados de la simulación realizada muestran que aquellos procesadores que deben recibir un dato acaban primero, mientras que aquellos procesadores que no deben recibir ningún dato tardan más tiempo en finalizar por esperar una señal de confirmación.

Una ventaja importante de esta simulación es que el patrón de conexiones del R-Mesh es un bosque. Este hecho hace que este R-Mesh se pueda simular en un LR-Mesh en $O(1)$ u.t.

VII.2 Trabajo futuro

Como trabajo futuro, se realizarán más experimentos y en mallas más grandes y se determinará formalmente el tiempo de ejecución promedio de este algoritmo. Además se analizará el caso para cuando el DR-Mesh tiene ciclos; para encontrar después el poder computacional del DR-Mesh acíclico con respecto al cíclico.

Otra posible línea de investigación es realizar la simulación del DR-Mesh sobre el LR-Mesh usando patrones de conexiones de algoritmos reales y no patrones aleatorios como los que se usan en los experimentos descritos en esta tesis.

Bibliografía

[Ben-Asher *et al.*, 1992] Ben-Asher, Y., Gordon, D. & Schuster, A. 1992. "Optimal Simulations in Reconfigurable Arrays". Technical Report no. 716, Technion Israel Institute of Technology.

[Ben-Asher *et al.*, 1994] Ben-Asher,Y., Lange,K.-J., Peleg, D. y Schuster, A. 1994. "The complexity of reconfiguring network models". Manuscript.

[Ben-Asher *et al.*, 1995] Ben-Asher, Y., Lange, K.-J., Peleg, D., y Schuster, A. 1995. "The Complexity of Reconfiguring Network Models". Info. and Comput. 121(1):41-58 p.

[Ben-Asher *et al.*, 1991] Ben-Asher, Y., Peleg,D., Ramaswami, R., & Schuster, A. 1991. "The Power of Reconfiguration". J. Parallel Distrib. Comput. 13:139-153 p.

[Ben-Asher y Schuster, 1991] Ben-Asher, Y., Schuster, A. 1991. "Data gathering on reconfigurable networks". Technion Israel Institute of Technology, Technical Report 696.
[Bondalapati y Prasanna, 1997] Bondalapati, K., & Prasanna, V. K. 1997. "Reconfigurable Meshes: Theory and Practice". In Reconfigurable Architectures Workshop,RAW'97.

[Chung, 1996] Chung, K. L. 1996. "Image Template Matching on Reconfigurable Meshes". Parallel Proc. Letters, 6(3):345-353 p.

[Duato *et al.*, 1997] Duato, J., Yalamanchili, S., Ni, L. 1997. "Interconnection Networks An Engineering Approach". IEEE Computer Society, Los Alamitos, California. 18-19 p.

[ElGindy y Wetherall, 1997] ElGindy, H., y Wetherall, L. 1997. "A Simple Voronoi Diagram Algorithm for a Reconfigurable Mesh". IEEE Trans. Parallel Distrib. Systems, 8:1133-1142 p.

[Fernández-Zepeda, 1999] Fernández-Zepeda, J. A. 1999. "Scaling Simulations of Reconfigurable Meshes". tesis doctoral, Louisiana State University, Baton Rouge.

[Fernández-Zepeda *et al.*, 1999] Fernández-Zepeda, J. A., Vaidyanathan, R., Trahan, J. L. 1999. "Improved Scalability Simulations of the General Reconfigurable Mesh". Proc. 6th Reconfigurable Architecture Workshop. LNCS 1586:616-624 p.

[Gál, 1995] Gál, A. 1995. "Semi-unbounded fan-in circuits". Boolean vs. arithmetic, The University of Chicago.

[JáJá, 1992] JáJá, J. 1992. "An introduction to Parallel Algorithms". Addison-Wesley Publishing Co. Primera edición. New York. 576 pp.

[Jang *et al.*, 1997] Jang, J.-w., Nigam, M., Prasanna, V. K., y Sahni, S. 1997. "Constant Time Algorithms for Computational Geometry on the Reconfigurable Mesh". IEEE

Trans. Parallel Distrib. Systems, 8:1-12 p.

[Jang y Prasanna, 1995] Jang, J.-w., y Prasanna, V. K. 1995. "An Optimal Sorting Algorithm on Reconfigurable Mesh". J. Parallel Distrib. Comput., 25(1): 31-41 p.

[Jang y Prasanna, 1997] Jang, J.-w., y Prasanna, V. K. 1997. "An Optimal Multiplication Algorithm on Reconfigurable Mesh". IEEE Trans. Parallel Distrib. Systems, 8(5):521-532 p.

[Kapoor, 1993] Kapoor, A. 1993. "Implementation of Communication Intensive Algorithms on Reconfigurable Mesh Arquitectures". Bachelor of Technology, IIT Delhi. 128 pp.

[Karp y Ramachandran, 1990] Karp, R. M., y Ramachandran, V. 1990. "Parallel algorithms for shared-memory machines". Handbook of Theoretical Computer Science, 895-941 p.

[Li y Maresca, 1989] Li, H., y Maresca, M. 1989. "Polymorphic-torus network". IEEE Trans. Comput., 38(9):1345-1351 p.

[Li y Stout, 1991] Li, H., y Stout, Q. F. 1991. "Reconfigurable SIMD Massively Parallel Computers". IEEE Proceedings, 79(4):429-443 p.

[Miller y Prasanna-Kumar, 1993] Miller, R., Prasanna-Kumar, V. K. 1993. "Parallel Computations on Reconfigurable Meshes". IEEE Transactions on Computers. 42(6):678-692 p.

[Miller *et al.*, 1988] Miller, R., Prasanna-Kumar, V. K., Reisis, D., y Stout, Q. F. 1988. "Image computations on reconfigurable VLSI arrays". in Proc. IEEE Comput. Soc. Conf. Comput. Vison Pattern Recognition, 925-930 p.

[Miller *et al.*, 1988b] Miller, R., Prasanna-Kumar, V. K., Reisis, D.,y Stout, Q. F. 1988. "Meshes with reconfigurable buses". in Proc. Fifth MIT conf. Advanced Res. VLSI, 163-178 p.

[Miller *et al.*, 1993] Miller, R., Prasanna-Kumar, V. K., Reisis, D., y Stout, Q. F. 1993. "Parallel Computations on Reconfigurable Meshes". IEEE Trans. Comput., 42(6):678-692 p.

[Miller y Stout, 1996] Miller, R., y Stout, Q. F. 1996. "Parallel Algorithms for Regular Arquitectures". Meshes and Pyramids. MIT Press.

[Moreira, 1997] Moreira, A. 1997. "Reconfigurable Meshes with Reconfigurable Switches". International Conference PDPTA97.

[Nakano, 1994] Nakano, K. 1994. "Efficient Summing Algorithms for a Reconfigurable Mesh". Proc. IPPS 94 Workshop on Reconfigurable Arquitectures.

[Schuster, 1991] Schuster, A. 1991. "Dynamic Reconfiguring Networks for Parallel Computers". Algorithms and Complexity Bounds. PhD thesis, Hebrew University, Jerusalem, Israel.

[Sidhu *et al.*, 1997] Sidhu, R. P., Bondalapati, K., Choi, S., y Prasanna, V. K. 1997. "Computation Models for Reconfigurable Machines". International Symposium on Field-Programmable Gate Arrays.

[Stockmeyer y Vishkin, 1984] Stockmeyer, L., y Vishkin, U. 1984. "Simulation of parallel random access machines by circuits". SIAM Journal on Computing, 13(2):409-422 p.

[Stout, 1992] Stout, Q. F. 1992. "Ultrafast Parallel Algorithms and Reconfigurable Meshes". In Proc. DARPA. Sortware Technology Conference 1992, 184-188 p.

[Thiruchelvan *et al.*, 1993] Thiruchelvan, R. K., Trahan, J. L., y Vaidyanathan, R. 1993. "On the Power of Segmenting and Fusing Buses". Proc. of international Parallel Processing Symposium, 79-83 p.

[Trahan *et al.*, 1998] Trahan, J. L., Bourgeois, A. G., y Vaidyanathan, R. 1998. "Tighter and Broader Complexity Results for Reconfigurable Models". Parallel Proc. Letters, 8:82-94 p.

[Trahan *et al.*, 1997] Trahan, J. L., Vaidyanathan, R., Bourgeois, A. G. 1997. "LRN Simulation of RN and RN Simulation of DRN". Manuscript, June 13.

[Wang y Chen, 1990] Wang, B. F., y Chen, G. H. 1990. "Constant Time Algorithms for the Transitive Closure and Some Related Graph Problems on Processor Arrays with Reconfigurable Bus Systems". IEEE Transactions on Parallel and Distributed Systems, 1(4):500-507 p.

Vita

José Antonio Cárdenas Haro se graduó en la carrera de ingeniería electrónica en el Instituto Tecnológico de Los Mochis, en diciembre de 1995; trabajó posteriormente por tres años y medio en la industria. Le fue otorgado el grado de Maestro en Ciencias en el CICESE en noviembre del 2001, en el área de Ciencias de la Computación, en la especialidad de cómputo paralelo. Sus intereses incluyen sistemas operativos Unix, análisis y diseño de algoritmos, redes neuronales, cómputo paralelo y distribuido, y arquitecturas paralelas de computadoras.